UNIVERSIDAD CARLOS III DE MADRID

Escuela Politécnica Superior.
Ingeniería Técnica Informática de Gestión.

PROYECTO FIN DE CARRERA

TRANSWIKI

Traducción multilingüe de palabras usando como recurso léxico-semántico Wikipedia.

Rubén Rodríguez Barroso

Junio de 2009

1. Contenido.

2. Índice de tablas.

3. Índice de Figuras.

4. Introducción.

En los últimos años se está produciendo un cambio en los sistemas de información globales, gracias al esfuerzo desinteresado de miles de contribuyentes que aportando su pequeño granito de arena logran hacer construir enormes castillos de información.

Pioneros en este tipo de trabajos fueron los sistemas Linux, construidos a partir del esfuerzo de miles de programadores, que aportaban su trabajo para lograr construir sistemas muy estables y seguros. Hoy en día, existen multitud de proyectos colaborativos que aportan a la humanidad una fuente incalculable de datos ó servicios de manera gratuita, como podemos hablar de Wikipedia, Wiktionary, Wikiquote ó bien, cualquier otro sistema Wiki.

Por otro lado, la informática avanza a tal velocidad que disponemos de muchos algoritmos ó programas de excelente calidad ya realizados y dispuestos para ser utilizados. Por todo ello, en múltiples ocasiones el principal activo del sistema informático no resulta de la arquitectura, sino de la información alimentada en el mismo. Es objetivo prioritario de cualquier campo, disminuir costes en proyectos, para ello, es interesante que los ingenieros de hoy estén capacitados para reutilizar los datos y hacerlos funcionales para el nuevo sistema.

Amplios son los sistemas de bases de datos que nos permiten guardar de manera optima y sencilla nuestros datos, muchos además, los algoritmos de búsqueda implementados que los hacen realmente eficientes. La idea principal de este proyecto versa en, partiendo de esos activos que están a nuestro alcance como son sistemas de bases de datos y la información contenida en la Wikipedia, construir un sistema de traducción que nos permita en un corto periodo de tiempo, **preparar de manera sencilla nuestros equipos para trabajar con un optimo y eficiente traductor** con versatilidad para trabajar con distintas aplicaciones.

5. Acrónimos.

- **JWPL :** Java Wikipedia Library.
- **JWTKL:** Java Wiktionary Library.
- **PLN :** Procesadores del lenguaje natural.
- **API:** Aplication programming interface.
- **CKB:** Collaborative Knowledge Base.
- **LKB:** Linguistic Knowledge Base.
- **SSH :** Secure Shell.
- **XML:** Extensible markup lenguage.
- **PHP:** Hypertext preprocessor.
- **SQL:** Structured Query Language.
- **IIS:** Internet Information Server.
- **HTML:** Hypertext markup lenguage.
- **Langcode:** Language code.

6. Estado de la cuestión.

6.1. Wikipedia.

6.1.1. Introducción.

Wikipedia es un proyecto de la fundación Wikimedia para la creación de una enciclopedia libre y políglota. Actualmente cuenta con más de 12 millones de artículos y está editada en más de 265 idiomas. A pesar de tener una historia apasionante, no se procederá a explicar estos datos en este estudio, sino que vamos a intentar mostrar la arquitectura software, hardware y de base de datos que presenta este sistema.

6.1.2. Arquitectura Lógica de la Wikipedia.

La arquitectura lógica de la Wikipedia se basa en su ítem principal que son las páginas, estas páginas son creadas por esfuerzo colaborativo de usuarios desinteresados y contienen los datos principales de una entrada de una enciclopedia. Para darle más información al sistema y accesibilidad, las páginas se agrupan en Categorías, pudiendo una misma página pertenecer a distintas categorías ó bien agrupar en una categoría distintas páginas.

En torno a las páginas se agrupan los restantes elementos que componen la Wikipedia, como son los usuarios que generan el contenido, las imágenes ó archivos multimedia que añaden información a las páginas, los enlaces externos e internos de las páginas, etc.

6.1.3. Arquitectura del servicio

Brevemente vamos a realizar un pequeño comentario sobre la arquitectura Hardware que soporta este sistema. Actualmente Wikipedia se ejecuta sobre clusters de servidores distribuidos por todo el mundo en máquinas Ubuntu Linux principalmente y algunas maquinas Solaris. Cuenta con más de 300 servidores en Florida, 26 en Ámsterdam y 23 en Corea.

Está diseñado usando programación por capas donde se separa la lógica de negocios de la lógica de diseño.

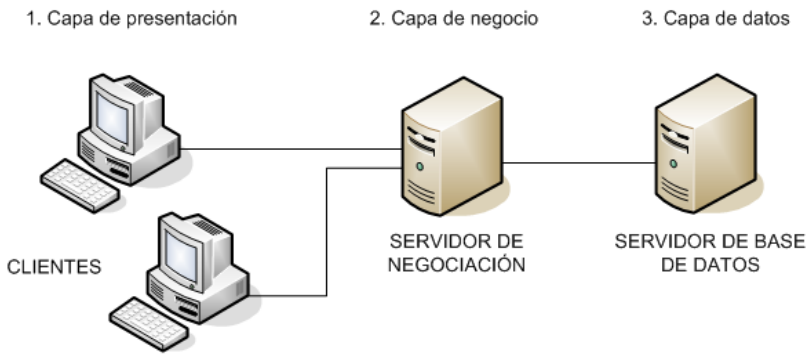

Figura 1- Programación por capas.

La separación en distintos servidores de la Wikipedia se realiza en servidores de bases de datos distribuidos MySQL, encargados del almacenamiento y recuperación de los datos, servidores web Apache encargados de la gestión de la página Web, y servidores Proxy squid caché encargados de albergar en caché las peticiones más frecuentes y disminuir con ello los tiempo de acceso.

Las características Hardware de los equipos que lo soportan son variadas, pero mostramos en la siguiente tabla las características técnicas de algunos de ellos a modo de hacerse una idea de la capacidad de trabajo con la que se dispone.

Servidor Base de datos	Servidor Web	Servidor Caché
Rackform nServ A210S CPU: Dual AMD Opteron 248 (2.2 GHz, 1MB L2 Cache) RAM: 16GB (8 x 2GB) PC3200/DDR400 NIC: Dual 10/100/1000 Mbps NIC (Broadcom 5704) - Integrated Integrated SCSI: No Integrated SCSI PCI-X 1: Adaptec SCSI 2230SLP - Low Profile 2 Channel U320 RAID w/ BBU 7 x HDD : Seagate Cheetah 15K.4 73GB U320 15KRPM SCA SCSI	# 1U, e.g. Dell PowerEdge 1950 # 2x Intel Xeon Quad-Core E5320 # 4 GB 533 Mhz # 2x 250 GB 7200 RPM SATA drives, 3,5" # No RAID # No redundant power supply # DRAC	# 2U, e.g. Dell PowerEdge 2950 # 1x Intel Xeon Dual-Core 5130 # 16 GB 667 Mhz (8x2GB) # 8x 2,5" SAS 36 GB # No RAID # No redundant power supply # DRAC

Tabla 1 - Características Hardware equipos Wikipedia

El entorno Software con el que trabaja la Wikipedia está basado en MediaWiki, que es un motor PHP para Wiki de software libre y puede ser utilizado en servidores Web Apache ó IIS, y servidores de base de datos MySQL y Postgre. MediaWiki resulta ser un simple intermediario entre la información contenida en la base de datos y el usuario final que realiza consultas, digamos de una manera

sencilla que es un frontend para Wiki, ya que también se utiliza en otros proyectos de la fundación Wikimedia como es Wiktionary.

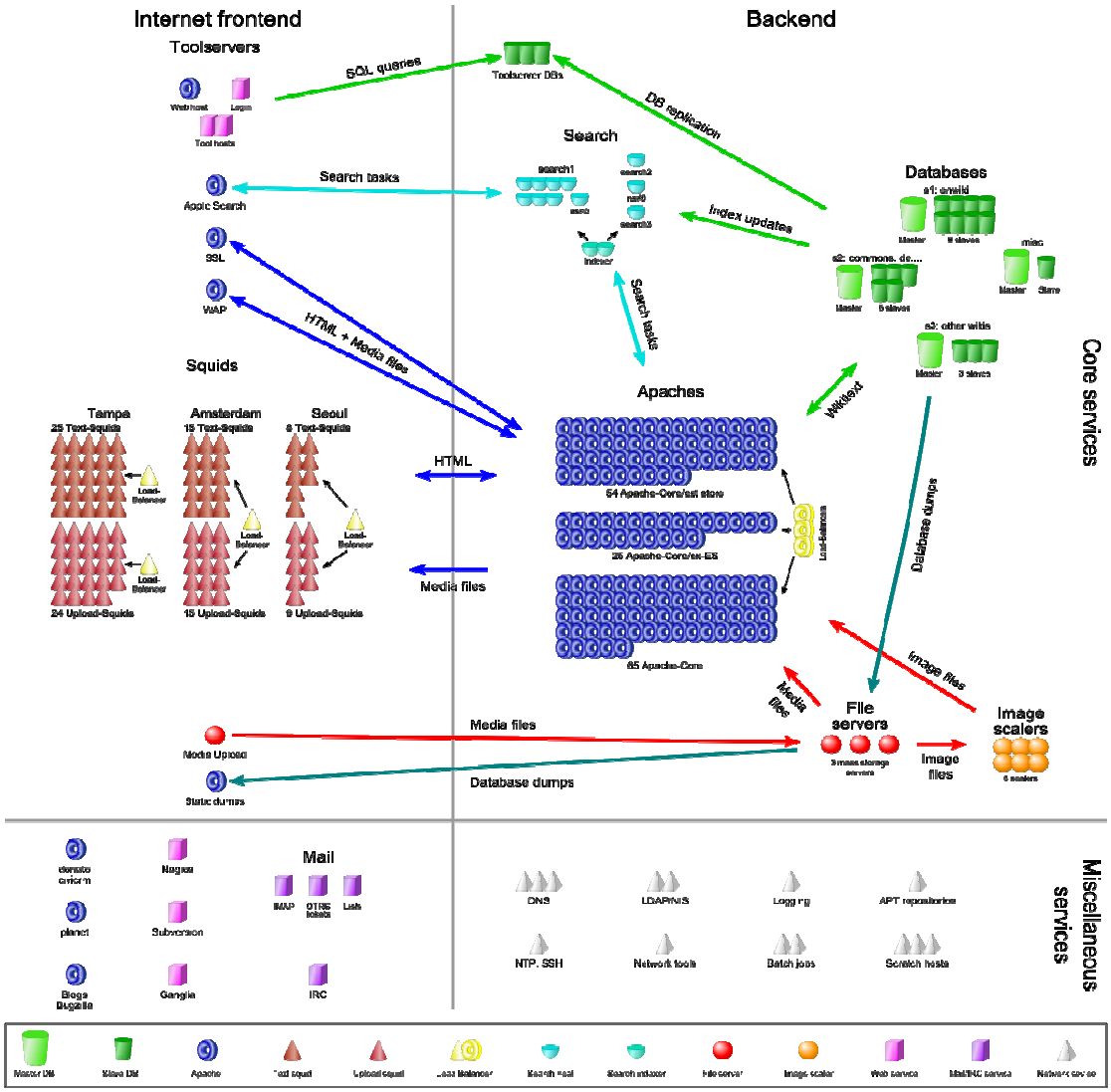

Figura 2- Arquitectura software y Hardware Wikipedia

6.1.4. Arquitectura de Base de datos.

La arquitectura de la base de datos es una parte muy importante de nuestro proyecto, ya que resulta de vital importancia conocer como y donde están ubicados los datos, para poder acceder a ellos de manera efectiva y transformarlos según nos interese.

La base de datos de la Wikipedia cuenta con ocho conjuntos de tablas con diferente funcionalidad, se encuentran por un lado las tablas con información referentes a los usuarios que son, `user`, `watchlist`, `user_group`, `user_newtalk`. La tabla que referencia a ip's bloqueadas llamada `ipblocks`. El conjunto de tablas que guardan estadísticas y accesos, llamadas `site_Stats`, `hitcount` y `logging`. Las tablas que se utilizan para pruebas de analizadores, `testitem`, `testrun` .Cuatro tablas que almacenan el distinto contenido multimedia, como son imágenes, archivos, videos, llamadas `filearchive`, `image`, `oldimage`, `imagelinks`. Existe además un conjunto de tablas que guarda información variada como puede ser

información de páginas externas que enlazan con la Wikipedia, meta datos, información de los trabajos de apache, etc., estas tablas son `trackbaks`, `job`, `interwiki`. El conjunto de tablas que almacenan información temporal `objectcache`, `math`, `transcache`, `querycache`, `querycacheinfo`, `querycachetwo` . Para nuestro objetivo el conjunto más importante es el que guarda la información referente al contenido de las páginas, `recentchange`, `archive`, `text`, `revision`, `templatelinks`, `categorylinks`, `redirect`, `externallinks`, `searchindex`, `page_restritions`, `pagelinks`, y las más importantes para nuestro cometido que son las tablas `Page` que guardan el contenido de la página y la tabla `langlinks`, que guarda la información a las páginas de otros alineadas de otros idiomas.

En el documento en el anexo *ejemplo de página de la Wikipedia*, podemos encontrar una imagen de una página normal de la Wikipedia que indica la procedencia de la información dentro de la base de datos. Siguiendo los distintos colores que se muestran podremos comprobar

Asimismo en el documento que consta en el anexo como *estructura de la base de datos de Wikipedia*, podemos encontrar el esquema entidad relación referente a la base de datos de Wikipedia.

6.1.5. Acceso a la Información.

El acceso a la información contenido en la Wikipedia, se puede realizar de distintas maneras:

- Mediante la página Web de la Wikipedia.
- Mediante rastreadores de datos ó robots, que acceden a la Wikipedia como si fuesen usuarios normales, muy utilizados para tareas de mantenimiento.
- Mediante la descarga de los DUMPS de Wikipedia, desde la página http://download.wikimedia.org/.

6.2. Wikipedia Optimizada JWPL-JWKTL

6.2.1. Introducción.

Actualmente, la Web está experimentando un mayor cambio a medida que mas y más personas están activamente contribuyendo en los contenidos disponibles en la llamada Web 2.0 .

Algunos de estos rápidos crecimientos, por ejemplo los que están desarrollando Wikipedia o Wiktionary, nos da el potencial de poder utilizarlos como un novedoso recurso léxico y semántico, debido principalmente a su creciente tamaño.

En particular, el potencial de la Wikipedia como una base de conocimientos léxicos y semánticos ya ha empezado a ser explorado en tareas del procesamiento del lenguaje natural (PLN), como categorización de textos, extracción de información, recuperación de información, respuesta a preguntas, calculo de las relaciones semánticas ó en el llamado reconocimiento de entidades.

Todas este tipo de tareas de PLN, requieren una información léxico y semántica fiable que normalmente proviene de bases de datos con conocimiento lingüístico como Word Net, ya que estas incluyen una serie de API's que permiten una sencilla integración con nuestras aplicaciones. Sin embargo, Wikipedia hasta ahora, ha carecido de este tipo de soportes que han significado un impedimento notable para las aplicaciones de PLN. JWPL y JWKTL es una API en Java para Wikipedia y Wiktionary que nos permitiese **acceder a la información de manera estructurada y sencilla desde nuestras aplicaciones**.

Por su lado, Wikipedia y Wiktionary son dos instancias de bases de conocimiento que son construidas de manera colaborativa por voluntarios no profesionales a través de la Web. A partir de ahora las vamos a llamar bases del conocimiento colaborativas (Collaborative Knowledge Base - CKB), a contraposición de las bases del conocimiento lingüístico (Linguistic Knowledge Base - LKB).

6.2.2. Comparación con LKBs.

La principal y más notable diferencia entre ambas entidades, es principalmente, que los **LKBs son construidos y mantenidos por profesionales del sector Lingüístico**. Por ello los LKBs tienen que seguir un modelo teórico más fuerte y llevar un consiguiente procedimiento estricto de prueba. La construcción de un sistema menos estricto como los CKBs, conlleva sus ventajas, como pueden ser: 1) **los CKBs suelen ser de uso gratuito** 2) **Al ser un proceso colectivo se actualizan y mejoran prácticamente a diario** 3) **los CKBs populares son mucho mas amplios y contienen más información** 4) **Incluyen información y enlaces sobre múltiples idiomas, y a idiomas que debido a su pequeño tamaño LKBs no contienen** 5) Suelen estar escrito en un lenguaje más coloquial que facilita notablemente su utilización para PLN.

6.2.3. Trabajo Relativo.

La manera más sencilla de obtener información de la Wikipedia es a través del buscador de la propia página Web. Sin embargo, este procedimiento no es apropiado para el acceso automático a los artículos a través de una aplicación.

El modulo de Perl WWW::Wikipedia (Summers, 2006), ofrece un simple medio para obtener artículos de la Wikipedia a través de querys programadas, sin embargo, este sistema genera una excesiva carga de datos para los servidores de la Wikipedia, que cuando se usa a gran escala lo convierte en inoperativo, por ello, está desaconsejado su uso por la fundación Wikimedia.

Otros enfoques que se han realizado a través de rastreadores Webs y por tanto no se adaptan a las necesidades de PLN son "the Wikipedia bot frameworks" (disponible para diferentes lenguajes de programación como python ó java) que puede ser usado para crear pequeños programas llamados robots que actúan en nombre de un usuario normal y que sirven para tareas de mantenimiento. "The Wiki Gateway tool box", una API unificada para interactuar con una amplia variedad de motores Wiki (Shanks, 2005).

El rastreo puede evitarse ejecutando un servidor propio disponible con los Dumps de Wikipedia. Esto da un mejor, pero aún insuficiente resultado, debido a la sobrecarga que produce el uso de un servidor Web para la petición de artículos. En este entorno una petición a Wikipedia implica una transferencia de peticiones de una aplicación a un servidor Web. El servidor Web entonces ejecuta un script php que accede a la base de datos de Wikipedia, y la base de datos devuelve el contenido del artículo codificado usando marcadores Wiki que posteriormente el script PHP convierte en HTML. Finalmente, el servidor Web devuelve el HTML codificado de vuelta a la aplicación. Esto supone una sustancial sobrecarga que puede hacer a larga escala imposible el uso para tareas de PLN.

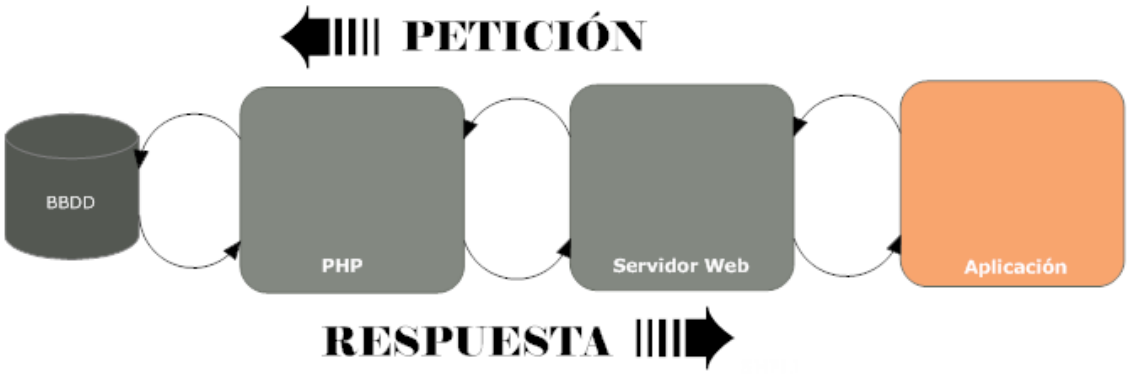

Figura 3- Ejemplo de WWW::WIKIPEDIA

Esta sobrecarga puede ser evitada si accedemos directamente a la base de datos. Por ejemplo, el modulo Perl Parse::MediaWikiDump (Riddle, 2006) examina los XML Dumps de la Wikipedia para devolver los artículos. Como los Dumps de la Wikipedia son demasiado grandes (más de 3 GB de información comprimida para la versión inglesa de la Wikipedia de febrero de 2008), el examinar los DUMPS de la Wikipedia tampoco es suficiente para el uso a alta escala de tareas de PLN (puede tomarse bastante tiempo en devolver un articulo solicitado). Adicionalmente, el tiempo que es requerido para devolver un artículo no es fácilmente predecible ya que depende de la posición del artículo dentro del archivo XML.

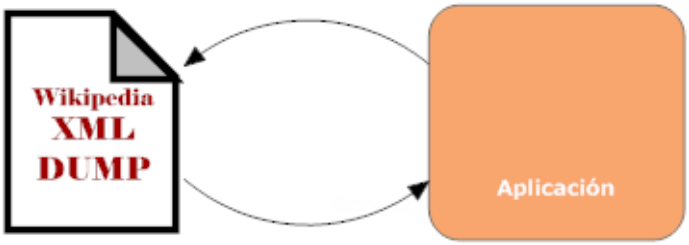

Figura 4 - Ejemplo de Perl Parse :: MediaWikiDump

WikiPrep (Gabrilovich, 2007) es un preprocesador que transforma el Dump XML de la Wikipedia en un XML optimizado que codifica información como heurísticas de redirección de los artículos. Sin embargo, como el archivo resultante está todavía en XML, WikiPrep acaba sufriendo el mismo resultado que Parse::MediaWikiDump.

En JWPL-JWTKL, se ha importado los XML Dump dentro de una base de datos. De esta forma, se pueden aprovechar las sotisficadas indexaciones que ofrece la base de datos y que garantizan un tiempo constante de devolución de cada artículo. Este enfoque es mejor que el las devoluciones del servidor Web, ya que solo envuelve a las peticiones a la base de datos y directamente devuelve los resultados a la aplicación.

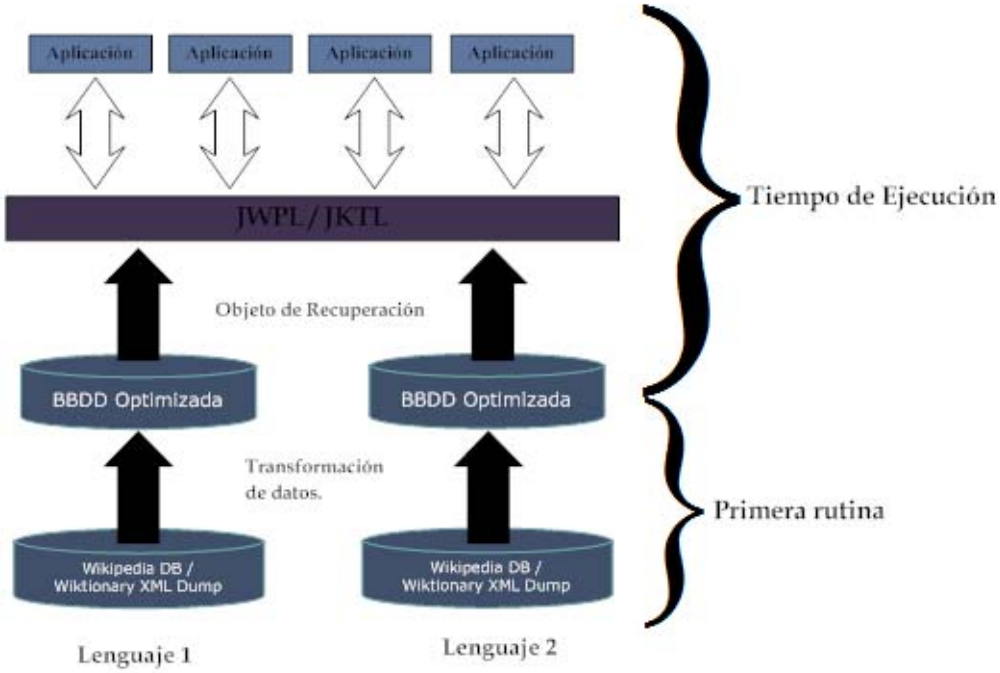

Figura 5 - Estructura JWPL /JKTL

6.2.4. Estructura JWPL

La estructura original de la Wikipedia está optimizada para realizar diariamente millones de búsquedas de artículos, buscándolos por palabra clave. Estas búsquedas son realizadas por millones de usuarios a través de todo el mundo. Sin embargo, un API diseñado para investigaciones PLN tiene que soportar un rango más amplio de accesos, incluyendo iteraciones sobre todos los artículos, la sintaxis de las consultas, así como un acceso eficiente a la información como vínculos, categorías y redirecciones. Por ello, JWPL opera con una base de datos optimizada que es creada en un primer momento a partir de los Dumps de la base de datos de la Wikipedia.

Las ventajas de la arquitectura de este sistema son: **1) Eficiencia en la computación a gran escala de tareas de PLN 2) resultados de la investigación estimables y reproducibles 3) un fácil interfaz de uso hacia la programación orientada a objetos.**

Los resultados experimentales que se pueden estimar y reproducir, son consecuencia de usar una base de datos especializada en contra posición con la Wikipedia online que es muy probable que se produzca distintas respuestas ante dos ejecuciones iguales de cierto experimento.

La eficiencia computacional es también una consecuencia del acceso a la base de datos usando mecanismos de indexación para una más rápida obtención de la información. Los datos de la base de datos son directamente mapeados hacia los objetos de Java usando "The Hibernate object-relational zapping Framework (Bauer and King, 2004). Esto también significa que JWPL no está restringido a utilizar una

cierta base de datos ya que puede funcionar con la mayoría de los más conocidos gestores de bases de datos.

El diseño de la orientación a objetos está centrado sobre los objetos: Wikipedia, Page, y Category. El objeto Wikipedia es usado para establecer la conexión con la base de datos y para Servir a los objetos Page y Category. JWPL soporta peticiones por clave ó por medio de un interfaz de búsqueda que permite para búsquedas con comodines, así como la recuperación de los subsectores de las categorías ó artículos dependiendo de los parámetros como el número de "tokens" en un artículo ó el número de vínculos entrantes. El objeto Wikipedia también permite iterar sobre los artículos, categorías, redirecciones y las páginas de desambiguación.

Un objeto Page representa ya sea un articulo normal de la Wikipedia, una redirección a un artículo ó una página de desambiguación. Cada objeto Page permite acceso al texto del artículo (con marcas de información ó texto plano), las categorías asignadas, los vínculos entrantes y salientes, así como todas las redirecciones que vinculan a ese artículo.

Los objetos category representan categorías de la Wikipedia y permiten acceder a los artículos dentro de esta categoría. Como las categorías en la Wikipedia forman un tesauro, un objeto category tambien proporciona los medios para obtener categorías padres e hijas, así como todos los hermanos y la colección recursiva de descendientes.

JWPL también proporciona un objeto CategoryGraph que por ejemplo permite encontrar el camino más corto entre dos categorías dadas.

6.2.5. Ejemplos de utilización en PLN.

Las APIs para el acceso a la Wikipedia y Wiktionary propuestas en este documento ya han sido puestas en funcionamiento para las búsquedas a larga escala de PLN, tales como el análisis y acceso a la estructura de la Wikipedia gráfica (zesch and gurevych, 2007), de relación de información semántica entre palabras (Zech et al, 2007), y recuperación de el texto de la entrada (Gurevych et al, 2007).

Cuando analizamos la estructura del grafico de categorías de la Wikipedia , las categorías asignadas a los artículos de Wikipedia son vistas como nodos en un gráfico dirigido, donde la subcategoría relación entre dos categorías se lanza como un arco que une los correspondientes nodos en el grafo. El objeto CategryGraph en JWPL ofrece los medios para recuperar los parámetros de un grafico como el diámetro, el coeficiente de grupo, el camino medio mas corto.

La estructura de los gráficos resultantes (según es definida por los parámetros del grafo) es indicativa de la posible actuación del gráfico basado en las aplicaciones PLN, por ejemplo, calculo de la semántica relacional entre palabras.

Estas tareas requieren recuperar el correspondiente articulo de la Wikipedia para cada palabra, y luego para calcular el mínimo camino entre las categorías de dos artículos. Sobre esta base, podemos encontrar fácilmente eficientes algoritmos para calcular las relaciones semánticas usando JWPL.

6.2.6. Conclusiones.

Recientemente, el recurso colaborativo Wikipedia fue descubierto como una base de conocimientos léxicos y semánticas que tiene el potencial de obtener mejores resultados en diversas áreas de PLN como categorización de texto, extracción de información, respuestas a preguntas, calculo de las relaciones semánticas o el reconocimiento del nombre de entidades. El proyecto de su joven hermana, Wiktionary, últimamente ha emergido como un recurso complementario a este. Hemos mostrado que estas bases de conocimiento colaborativo contienen conocimiento léxicos y semánticos que no son comúnmente encontrados en las bases del conocimiento lingüístico. La necesidad de un acceso programado apropiado a este conocimiento es evidente.

6.3. Tecnología.

6.3.1. MySQL.

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario. Desarrollado en su mayor parte en ANSI C, se encuentra bajo licencia GNU GPL y se dispone de su código abierto, a pesar de ser software propietario.

La arquitectura de MySQL está compuesta de varias capas, la primera es un administrador de conexiones de seguridad, la segunda capa es un analizador sintáctico de sentencias sql y la tercera capa está formada por distintos motores de almacenamiento, también llamados sistemas de tablas.

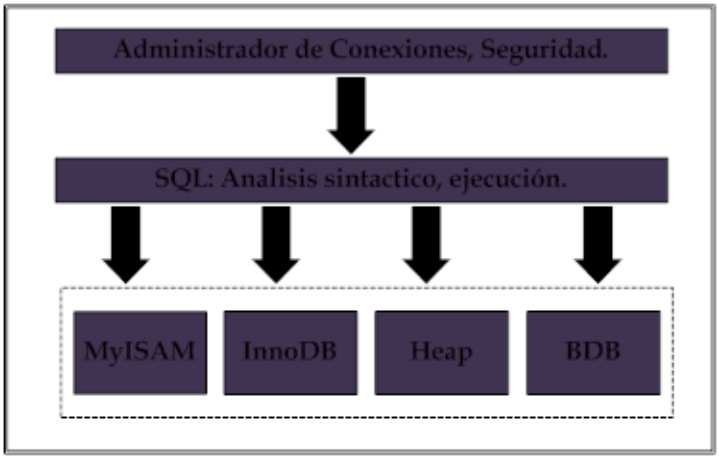

Figura 6 - Arquitectura de MySQL

En los sistemas gestores de bases de datos multiusuario y multihilo, uno de los principales problemas a tratar son los de los bloqueos, concurrencias y transacciones. En MySQL es la tercera capa la que se encarga de resolver estos problemas mediante los motores de almacenamiento, cada uno de ellos define una solución distinta para los mismos problemas.

Los motores de almacenamiento son los encargados del almacenamiento y posterior recuperación de los datos. Cada motor de almacenamiento presenta unas características especiales que lo hacen mas apropiados para unos sistemas u otros, a pesar de ello, podemos hacer uso combinado en nuestro modelo de los distintos motores. Por todo ello, es interesante comprender las principales diferencias y conocer las ventajas de cada uno.

Los motores de almacenamiento que implementa MySQL son MYISAM, InnoDB, Heap (en memoria), NDB. En la siguiente tabla se muestra un esquema general de la solución particular que se ofrece para cada problema

Atributo	MyISAM	Heap	BDB	InnoDB
Transaciones	No	No	Sí	Sí
Granuralidad de bloqueo	Tabla	Tabla	Página	Fila
Almacenamiento	Archivos Partidos	En memoria	Archivo único por tablas	Espacio de tablas
Niveles de aislamiento	Ninguno	Ninguno	Lectura confirmada	Todos
Formato Transportable	Si	N/A	No	Si
Integridad Referencial	No	No	No	Si
Clave Primaria con los datos	No	No	Si	Si
MySQL almacena los registros de datos	No	Si	Si	Si
Disponibilidad	Todas las versiones	Todas las versiones	MySQL-Max	Todas las versiones

Tabla 2 - Características motores MySQL

De estos cuatro sistemas, lo más utilizados y extendidos son MyISAM e InnoDB, los cuales expondremos brevemente:

Las tablas MyISAM son el primer sistema de tablas que se implementó para MySQL y es el sistema predeterminado. Las tablas MyISAM no son transaccionales, ni implementan integridad referencial, tampoco incluyen bloqueos de bajo nivel, sin embargo, permiten la indexación por texto completo, la compresión y otras características. Este sistema se caracteriza por resolver los problemas de concurrencia mediante bloqueos a las tablas completas, además permite implementar en las tablas campos fijos y dinámicos.

La principal característica de MyISAM es su velocidad en la recuperación de datos, su versatilidad en la indexación y tipos de tablas. Son especialmente apropiadas para sistemas con alta cantidad de datos, donde la integridad referencial no represente una necesidad obligatoria.

El almacenamiento de los datos se hace a través de conjunto de pares de ficheros para cada tabla, un fichero donde guarda los datos y otro fichero que guarda los índices.

El sistema de almacenamiento InnoDB, es el último sistema de tablas añadido a MySQL, el más refinado ya que añade características importantes. Los bloqueos los realiza a nivel de fila mediante cuatro niveles diferentes, con lo que para numerosas aplicaciones estos es una ventaja indispensable. Permite integridad referencial basándose en la clave ajena. Implementa un sistema rápido de búsqueda por claves indexadas, aunque la integridad referencial disminuye notablemente la velocidad en las inserciones. Es un sistema especialmente implementado para bases de datos con accesos concurrentes que requieran integridad referencial.

6.3.2. *Shell script.*

Shell script es un lenguaje de procesos para sistemas unix que incorpora ordenes para la línea de comandos de Linux. Este lenguaje de programación resulta muy versátil y nos ofrece la capacidad de automatizar procesos de sistema de una manera sencilla.

La utilización de shell script aporta ventajas con respecto a usar cualquier otro lenguaje de programación. Shell script no necesita ser compilado, sino que es un lenguaje de ordenes, por ello, te permite trabajar con mayor rapidez al no tener instalar en el sistema compiladores especializado. Asimismo, este lenguaje viene incorporado en todos los sistemas Unix , con lo que no es necesario ningún requerimiento especial. Por otro lado, al trabajar con shell script dependemos directamente de la línea de comandos de linux, por ello debemos ser especialmente cuidadosos ya que existen diferencias en algunos comandos entre las distintas versiones de este sistema y ello podría causar error en nuestro código.

7. Objetivo.

El objetivo de este proyecto versa en comprender y **conocer la información contenida en la Wikipedia y tratar dicha información de manera optima para construir un sistema de traducciones**.

Además de todo ello este sistema, debe ser lo suficientemente **rápido** como para traducir grandes cantidades de texto en tiempos razonables, asimismo también debe ser **adaptable a los distintos idiomas** que conforman la Wikipedia, es decir, debe estar capacitado para realizar traducciones entre los 261 idiomas que componen el sistema de Wikipedia.

Un objetivo muy importante para este proyecto, trata en que debe estar **integrado con el sistema de JWPL**, es decir, con la API para java de Wikipedia, de esta manera podremos ampliar dicha API para que además de las múltiples funciones que posee tenga la capacidad de realizar traducciones.

Para la consecución de dicho objetivo se establecieron diferentes metas a realizar:

- Implantación de un servidor propio de Wikipedia para tratar los datos.
- Estudio de la arquitectura de la página de Wikipedia.
- Implantación de un servidor de JWPL.
- Estudio de la arquitectura del sistema JWPL.
- Obtención y traducción de los datos para su uso en traducciones.
- Integración de los datos con la arquitectura de JWPL.
- Obtención de una arquitectura optima.
- Automatización del proceso.
- Pruebas.

La consecución de cada uno de estas metas ha sido sustancialmente importante para encaminar el proceso de desarrollo siguiente, y serán explicadas detalladamente en la memoria de trabajo.

8. Trabajo Realizado.

8.1. Instalación de MediaWiki en Ubuntu Linux.

8.1.1. Introducción.

Para poder estudiar el funcionamiento y esquema relacional que componen la estructura de la base de datos que alberga los datos de los artículos de la Wikipedia, es necesario la implementación de un Servidor Propio de Wikipedia que nos permita gestionar dicha base de datos con total libertad.

Para la implementación de la base de datos de Wikipedia, vamos a hacer uso de un motor para Wikis bajo licencia GNU programado en PHP y llamado MediaWiki así como, la restauración de las copias de seguridad de la base de datos de la Wikipedia que montaremos en nuestro propio sistema gestor.

8.1.2. Arquitectura de la aplicación.

MediaWiki para su puesta en funcionamiento, requiere tener instalado un servidor Web con capacidad para gestionar páginas Web desarrolladas con PHP, así como un sistema gestor de bases de datos relacionales.

En este proyecto para llevarlo a cabo hemos utilizado, un servidor Linux con la distribución Ubuntu Server 8.10 , servidor Web Apache2, PHP5 y un sistema gestor de bases de datos relacionales de software libre llamado MySQL. La configuración del Servidor se ha realizado por conexiones remotas a través de Secure Shell (SSH) y trabajando bajo la Shell de Linux.

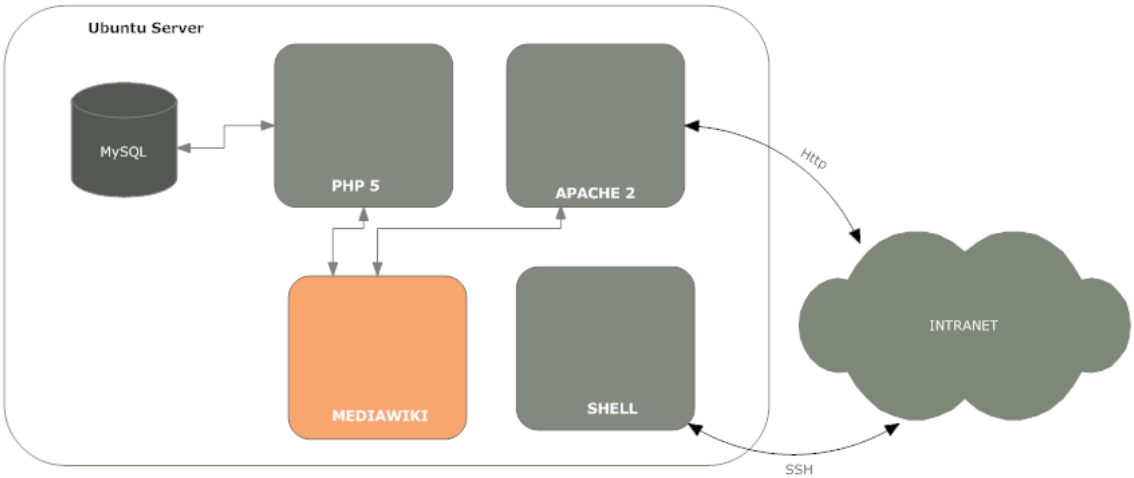

Figura 7. Arquitectura de MediaWiki

8.1.3. Procedimiento.

Se ha llevado a cabo la instalación y montaje de un servidor Ubuntu Linux basado en software libre y descargando la versión 8.10 Server de la página Web del proyecto Ubuntu.

Toda vez el servidor está operativo se ha montado un servidor Web Apache2 que nos permite visualizar páginas HTML desde otro equipo mediante la red.
Se añade la funcionalidad de lenguaje PHP, para que nuestro servidor Web Apache2 pueda dar servicio de páginas Web dinámicas.

Instalamos MySQL 5. Una vez esté configurado y operativo debemos crear un nuevo "schema" y un usuario con permiso total para dicho esquema. Ese usuario será el utilizado por MediaWiki para acceder a los datos de la aplicación que son de nuestro interés.

Se descarga e instala MediaWiki en el servidor y se configura con los parámetros adecuados para que pueda trabajar conjunto con la base de datos.
En este momento tenemos el servidor Wiki operativo, ahora nuestro procedimiento llevado a cabo es la descarga de los DUMPS de la base de datos de Wikipedia y su restauración en nuestra base de datos. Los DUMPS, vienen descritos en formato XML e invocamos unas librerías públicas de PHP que convierten el código XML en sentencias SQL válidas de forma automatizada para escribirlas en nuestra base de datos. Este proceso como se ha comentado anteriormente se realiza de forma automática y tiene una duración aproximada de tres días.

8.1.4. Manual de Usuario.

Para poder comprobar el funcionamiento de la correcta aplicación de nuestro servidor, tan solo necesitamos un navegador Web sobre el que ejecutaremos la siguiente dirección: http://localhost/wikipedia_es . Una vez abierto tan solo deberemos realizar una búsqueda adecuada en el cuadro correspondiente y el sistema automáticamente nos mostrará el contenido correspondiente a nuestra búsqueda.

8.1.5. Conclusiones.

La instalación de un servidor en MediaWiki requiere una formación especial en el mundo de los servidores Linux. Se requiere aproximadamente siete días de trabajo para la correcta instalación de un servidor de Wikipedia propio, con tan solo los datos de los artículos. Este tiempo puede aumentar notablemente si restauramos otros datos como comentarios, imágenes, etc. Ya que tan solo la restauración de las 31652 entradas que componen la Wikipedia Española suponen más de dos días de procesamiento en un computador normal.

8.2. Optimización de la base de datos de Wikipedia.

8.2.1. Introducción.

Tal y como se explica en el apartado anterior que hace referencia a la estructura de JWPL, es necesario adecuar la información de la base de datos de Wikipedia a una nueva base de datos optimizada que nos permita trabajar con la API de JWPL.

Para nuestro trabajo hemos utilizado una base de datos ya optimizada, como es la base de datos optimizada de la versión Inglesa. Sin embargo y al no disponer de la base de datos de Wikipedia optimizada de idiomas descendientes de la lengua latina, con los que nos parecía interesante trabajar para poder probar las traducciones, tuvimos que optimizarla por nuestra cuenta de la manera que se va a explicar en las siguientes líneas.

8.2.2. Procedimiento.

El proceso para optimizar la base de datos, es ante todo un procedimiento sencillo el cual necesita de unos importantes conocimientos técnicos sobre java para llevarlo a cabo.

Tenemos que hacer uso de una herramienta adjunta con JWPL e invocar su ejecución junto con los correspondientes DUMPS de la base de datos de Wikipedia, aquellos que hacen referencias a las páginas, los enlaces de las páginas y las categorías. Este proceso obtiene la información de los distintos DUMPS de Wikipedia y la estructura de una manera diferente para su uso en PLN.

A la hora de invocar esta operación resulta de suma importancia, aumentar la memoria asignada a los procesos JAVA en el sistema, ya que sino, este proceso no podrá realizarse.

Este proceso de transformación, se tiene que aplicar para todos los idiomas con los que se quiera implementar el sistema, para posteriormente poder hacer uso del mismo en la traducción.

Como ya se ha comentado anteriormente, es un proceso sencillo que conlleva aproximadamente algo más de una hora para poder transformar la Wikipedia y aproximadamente dos horas más para poder restaurar dicha información dentro de nuestra base de datos.

8.3. Restauración BBDD optimizada con JWLP

8.3.1. Introducción.

Para poder estudiar la estructura de la base de datos optimizada para el uso con JWPL, se ha procedido a crear una ampliación en nuestro sistema para introducir los datos optimizados de JWLP.

8.3.2. Arquitectura del Sistema.

La arquitectura del sistema tras la incorporación de el nuevo esquema, para uso con sería el mismo que el mostrado en la sección anterior añadiendo a nuestra base de datos MySQL el esquema nuevo.

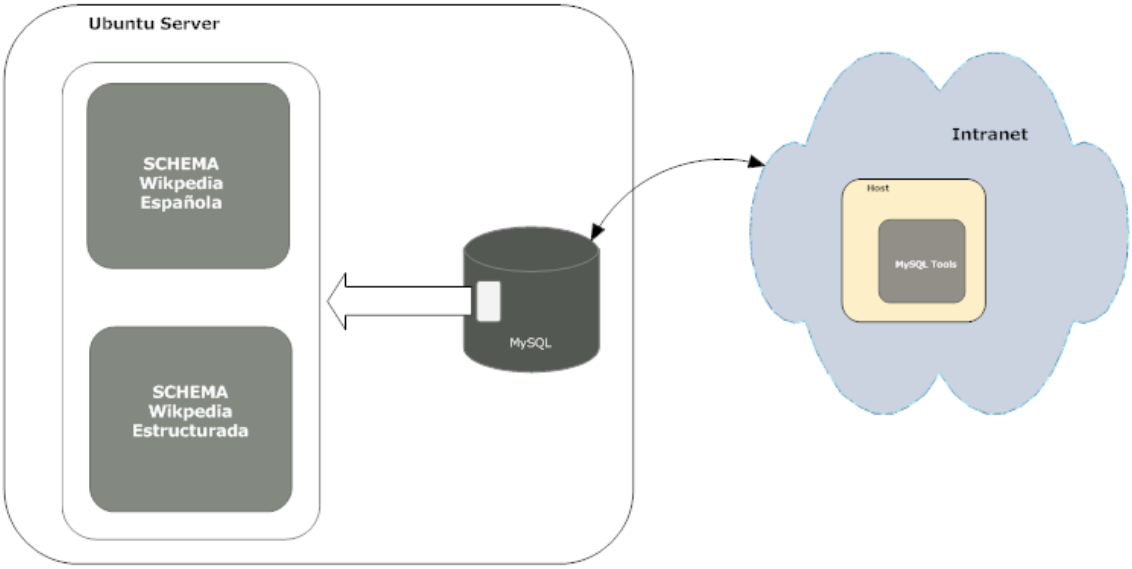

Figura 8 -Arquitectura con el esquema de Wikipedia Optimizada.

8.3.3. Procedimiento.

El procedimiento de restauración consiste en descargar de la Web el archivo con la copia de seguridad de la base de datos optimizada e invocar a MySQL para que ejecute las sentencias e introduzca debidamente los datos en nuestro sistema gestor de bases de datos.

8.3.4. Manual de Usuario

La utilización de este nuevo sistema se efectúa realizando consultas a la base de datos MySQL, bien sea, mediante lenguaje SQL ó bien utilizando la API especialmente desarrollada para este propósito Java-Based Wikipedia Library.

8.3.5. Conclusiones.

La restauración de la base de datos de la Wikipedia optimizada, es un proceso técnicamente muy sencillo, pero sin embargo, lleva un espacio temporal amplio, ya que la restauración de la base de datos requiere bastante tiempo para realizarse

En nuestro ejemplo, la restauración tardó unas 8 horas.

8.4. Estudio de la arquitectura de la base de datos de JWPL.

8.4.1. Introducción.

Para poder comprender mejor el funcionamiento del sistema y poder utilizarlo para nuestros intereses, es importante conocer el esquema relacional que compone JWPL. De esta manera podremos comprender mejor como integrar nuestra aplicación con este sistema.

8.4.2. Arquitectura del Sistema.

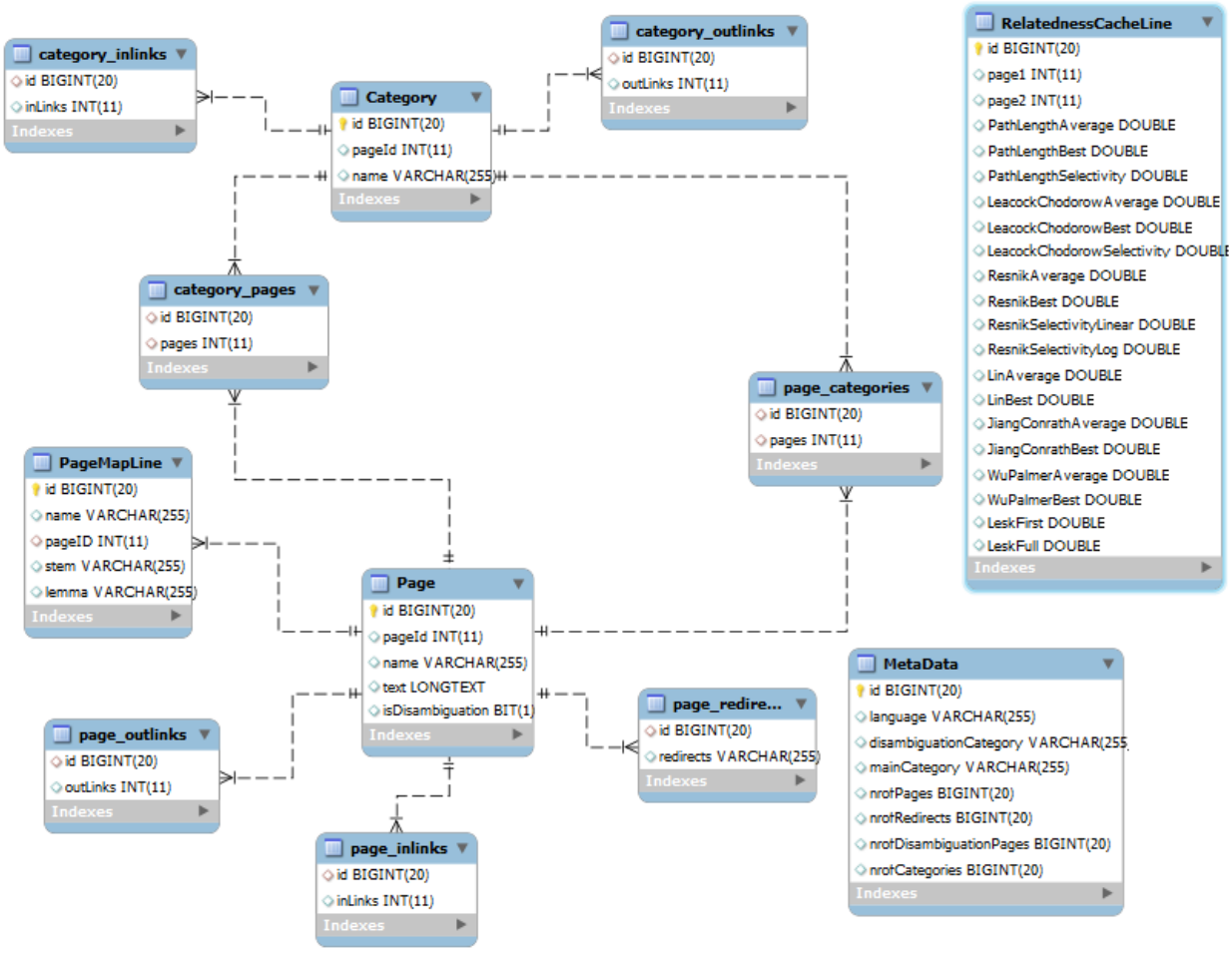

Figura 9 - Arquitectura de la BBDD de JWPL

La estructura tiene las siguientes utilidades:

- **`Page`:** Esta tabla contiene la información sobre las páginas. Contiene un campo identificador, su nombre y la información que contiene esa página. Se puede decir que es el núcleo central de la base de datos optimizada.
- **`Page_outlinks`:** Esta tabla contiene la información sobre los enlaces salientes de la página, contiene el id de la página y el id de la página referenciada. Esta tabla es muy útil para calcular la distancia entre dos palabras basándonos entre la distancias de enlaces entre una y otra.
- **`Page_inlinks`:** Contiene la información de todas las páginas que acceden a la página dada. Contiene los mismos campos que la tabla anterior y sin embargo su información es la inversa. Esta tabla nos puede dar una

información adicional sobre la importancia de la página que estamos tratando, es decir, contra mayor número de enlaces entrantes podemos suponer que mayor importancia debe tener esa página.

- **`Page_redirects`:** Contiene la información de los sinónimos de las páginas, es decir, aquellas páginas que redireccionan a una página dada, también es un valor que nos muestra la importancia de una página dada..
- **`Category`:** Contiene la información sobre las categorías. Las páginas pertenecen a categorías y las categorías a su vez pueden pertenecer a nuevas categorías.
- **`Page_categories`:** Asocia las páginas con sus respectivas categorías.
- **`Category_pages`:** Asocia las categorías con sus respectivas páginas.
- **`Category_outlinks`:** Esta tabla contiene la información sobre los enlaces externos que contiene cada categoría. Es un importante dado para calcular la importancia de la categoría, ya que contra más grande sea la categoría mas enlaces a páginas referenciadas tendrá.
- **`Category_inlinks`:** Esta tabla contiene los enlaces entrantes a una categoría dada.
- **`MetaData`:** Información de la propia Wikipedia con la que está trabajando la base de datos.
- **`PageMapLine`:**
- **`RelatednessCacheLine`:** Es una tabla que alberga información temporal para consultas de alineación de palabras en PLN.

8.4.3. Procedimiento.

El procedimiento seguido para estudiar la estructura de la base de datos ha sido realizar consultas y revisar la estructura de las tablas. Al estar diseñadas en MyISAM no contienen relaciones entre tablas y resulta difícil encontrar correspondencia entre los datos.

8.5. Traducción de datos.

8.5.1. Introducción.

Previo a la implementación del sistema, debemos obtener y traducir los datos necesarios para convertirlos en funcionales para nuestro objetivo de traducción. Debido a la alta cantidad de datos que manejamos, la traducción de los mismos deja de ser una tarea trivial y nos obliga a estudiar técnicas optimizadas que nos permitan solventar eficientemente nuestro problema.

Los datos de las traducciones para la Wikipedia que alimentan nuestro sistema, los obtenemos a partir de un DUMP de Wikipedia llamado `langlinks`.

`Langslinks` incorpora una estructura de tabla formada por un identificador principal de la página `ll_from` que es clave ajena de el `PageId` de la Wikipedia, `ll_lang` que representa el idioma de esa traducción y `ll_title` que representa el nombre de la página que traduce.

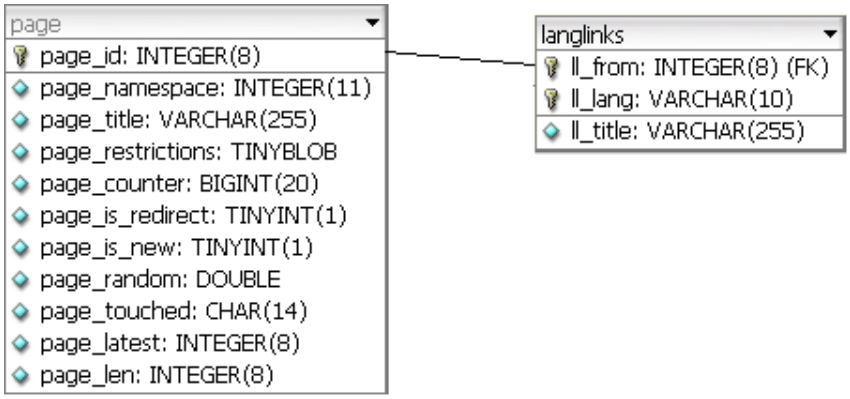

Figura 10- Esquema de Langlinks

La principal tarea a realizar es conseguir traducir esa información y optimizarla para poder usarla junto al sistema de JWPL en el trabajo del lenguaje natural. Como paso previo se ha realizado un estudio exhaustivo del significado de cada uno de los campos con los que se trabaja en la base de datos de JWPL . Se ha intentado traducir esa información con el pertinente dato de la base de datos de Wikipedia, encontrando correspondencias entre ambos sistemas.

El trabajo a realizar consiste primeramente en filtrar los datos del idioma que nos interesa. Posteriormente, debemos traducir `ll_from` por el correspondiente `ID` de la página del idioma origen, también tenemos que corresponder `ll_title` con el `ID` de la página del idioma destino. La siguiente imagen muestra el proceso de traducción de datos que se debe llevar a cabo.

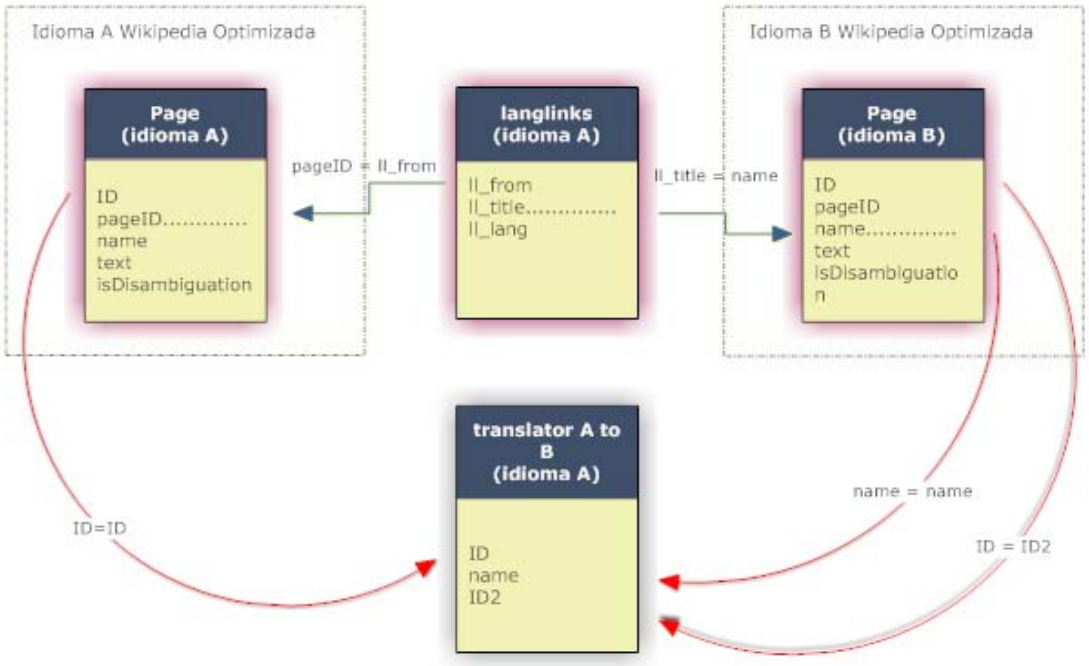

Figura 11 - Proceso de traducción global de datos

8.5.2. Traducción de datos.

8.5.2.1. Proceso teórico

Supongamos ahora un proceso de traducción de datos equivalente al mostrado anteriormente. Tenemos dos idiomas dados, un idioma A con N_A filas en la tabla "`Page`" y un idioma B con N_B filas en la tabla "`Page`", asimismo, se dispone de una tabla `langlinks` para el idioma A con N_{AL} filas y otra tabla `langlinks` para el idioma B con N_{BL} filas. La información se muestra indexada por 3 campos, los campos `ID` de las tablas `Page` y el campo ll_from de las tablas `langlinks`. Estimamos un tamaño medio de fila de T_A , T_B , T_{AL} , T_{BL} respectivamente para cada tabla y un tamaño de bloque de BL.

Con los datos dados, deseamos iniciar una primera traducción de la tabla `langlinks` del idioma A. Para traducir el campo `ll_from` por `ID`, debemos comparar el campo `PageID` de la tabla `Page` con el campo `ll_from` de `langlinks`, el campo `pageID` no está indexado por lo que deberemos realizar por termino medio **(N_{AL} x T_A / 2)** comparaciones. Esta consulta no se beneficia de memoria caché de consultas, pero sí de indexación, el sistema gestor de base de datos realiza un pequeño estudio tras el cual revisa las estrategia optima para realizar la consulta. En este caso podemos utilizar dos estrategias, la estrategia de escaneo normal y la estrategia de escaneo por índice, frecuentemente el sistema intentará utilizar índices ya que en transacciones de comparación es más rápido.

Para nuestro estudio, nos centraremos principalmente en los accesos a disco que supone el trabajo a realizar, ya que los tiempos de comparación de datos en memoria y trabajo de procesador pertinente se pueden considerar despreciables en tiempo en comparación con los accesos a disco.

Utilizando escaneo por índice, el sistema carga el índice en memoria, proceso que podemos calificar como despreciable debido a su pequeño tamaño. A la carga del índice le debemos sumar la carga en disco de la tabla `Page`:

$$F1 = (N_A /(BL / T_A))$$

Una vez que la comparación resulta positiva debe acceder a la tabla `langlinks`, por lo que debemos sumar los accesos a disco que supone la carga de datos de la tabla `langlinks` del idioma A,

$$F2 = (N_{AL} /(BL / T_{AL}))$$

En total tenemos

$$F4= F1+F2= (N_A /(BL / T_A)) + (N_{AL} /(BL / T_{AL}))$$

accesos a disco si realizamos esta consulta con un índice.

Utilizando escaneo normal y suponiendo un hipotético caso de memoria ilimitada, solo se accedería a disco para cargar las dos tablas que producen la comparación, es decir, la tabla `langlinks` y la tabla `page` para el idioma A. Para cargar en memoria la tabla `langlinks` de A necesitamos `[F2]` accesos a disco y consecuentemente `[F1]` para cargar la tabla `Page`, en total, `[F4]` accesos a disco. Sin embargo, si nos acercamos más a la realidad suponiendo un sistema con memoria bastante limitada, los accesos a disco se incrementan notablemente, ya que por cada bloque leído de `langlinks` deberemos leer todos los bloques de la tabla `Page` y comparar entre sí las filas incluidas. Por todo ello, si tenemos `[F2]` accesos a disco para `langlinks` y `[F1]` para `page`, tendremos un total de accesos de

$$F5= F1 \times F2=(N_{AL} /(BL / T_{AL})) \times (N_A /(BL / T_A)).$$

Como aproximación a un caso real de escaneo sin índice, sería demasiado arriesgado suponer una memoria ilimitada ó por el contrario suponer una memoria bastante limitada, por ello, se puede estimar con una buena aproximación un termino medio de ambos casos, con lo cual tendremos un número de accesos aproximado de

$$F6=(F4+F5)/2=((N_{AL} /(BL / T_{AL})) \times (N_A /(BL / T_A))) + ((N_{AL} /(BL / T_{AL})) + (N_A/(BL / T_A)) / 2)$$

para realizar la traducción del `pageID` con el `ll_from` y obtener el `ID`.

Para la realización de la traducción de `ll_title` por el campo `name`, estamos en una situación parecida a la anterior, por lo que necesitaríamos cargar en memoria la tabla `langlinks [F2]` y posteriormente cargar la tabla `page` del idioma B

$$F7 = (N_B /(BL / T_B))$$

De manera que para realizar la traducción de `ll_tite` y por el campo `name` y obtener el `ID2` tendríamos un total de accesos a disco de:

$$F8= ((F7 \times F2) + (F7 + F2)) / 2$$

En general, para realizar la traducción anterior en una sola transacción, debemos sumar el número de accesos de ambas consultas y añadirle los accesos de escritura de los resultados en disco. En total `([F6] + [F8])` sí ejecutamos la consulta sin índice ó `([F8] + [F4])` sí ejecutamos la consulta con índice. Por todo ello, ejecutar esta consulta sin fragmentar, y suponiendo valores para N elevados, la

podemos prever como inviable para el sistema informático que nosotros manejamos.

Una alternativa para solventar el problema, es dividir la consulta anterior en diferentes consultas más pequeñas usando tablas temporales.

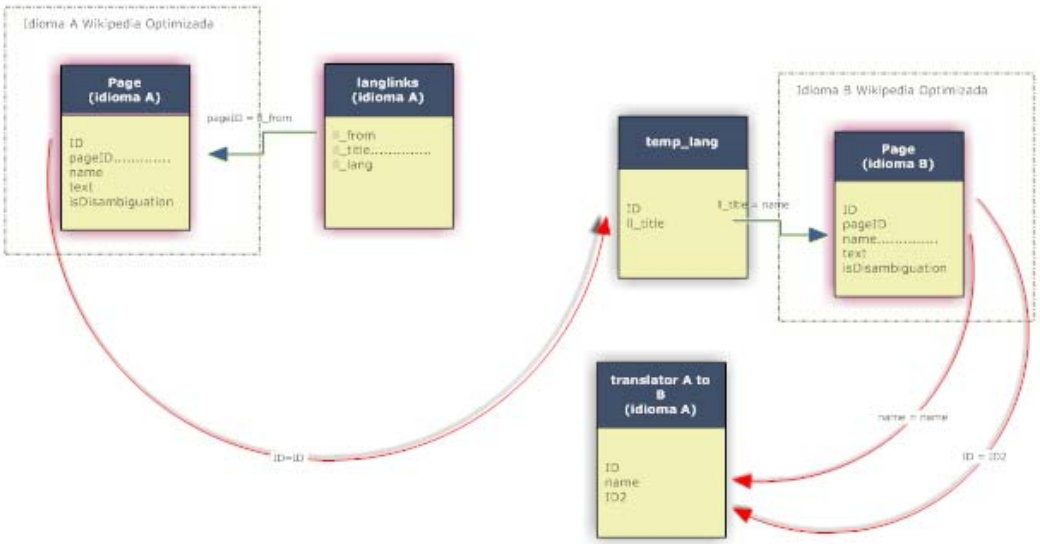

Figura 12 -Proceso fragmentado

Definamos ahora los siguientes pasos como un segundo proceso. En este segundo proceso creamos una tabla temporal para descomponer la consulta en dos pasos. Esta tabla temporal llamada `temp_lang` contendrá los campos `ID`, `ll_title` y trabajaremos con los campos filtrados por el idioma que nos interesa. En el primera paso para rellenar esta tabla comparamos los campos `ll_from` de la tabla `langlinks` con el campo `pageID` de la tabla `page` del mismo idioma y obtendremos el respectivo campo `ID` y `ll_title` para rellenar la tabla. Esta consulta se puede ejecutar usando el escaneo por índice explicado anteriormente, con lo que tendremos `[F4]` accesos a disco. El siguiente paso es traducir el campo `ll_title`, por el campo `name` y obtener así el `ID2` deseado de la tabla final. Como hemos visto anteriormente y al no tener ningún índice para la consulta, el número total a discos para este segundo paso es de `[F8]` para realizar la traducción del `pageID` con el `ll_from` y obtener el `ID2`. Esta segunda consulta para campos muy elevados de N, también resulta inviable, con lo que la fragmentación expuesta tampoco resulta viable.

Para solventar el problema anterior, podemos introducir un índice para el campo `name` de la tabla `page`. La comparación del campo `ll_title` y `name` se realiza con una estrategia de escaneo por índice, por lo que los accesos a disco disminuyen notablemente, con este nuevo sistema tendríamos que cargar en memoria la tabla `lang_temp` que es aproximadamente la mitad de la tabla `langlinks`, F3=(`[F2]/2`), cargamos también el índice que por su poco tamaño consideramos despreciable, y añadirle el acceso de las N_{AL} =52258 filas que coinciden y se deberán obtener de la tabla `Page`, en total

$$F9= F3+(N_{AL}/(BL/T_B)) = (N_{AL}/(BL / T_{AL}))/2 + (N_{AL} /(BL / T_B)).$$

En conclusión el estudio teórico nos indica que el proceso debe ser realizado utilizando una tabla temporal y añadiendo un índice para el campo `name` de las tablas `Page`. El índice por el campo `name`, es necesario para posteriori el sistema de

traducción que queremos implementar. Si el campo `name` no está indexado, por cada consulta a traducir el sistema debería cargar toda la tabla `page` en memoria y comparar los campos uno por uno con nuestra página a buscar, un total de accesos de `[F1]+ 1` , para la tabla A. Al añadir el campo `name` indexado, los accesos a cada consulta disminuyen **a log(N_A)/ log(BL /(3 X 2)/ (longitud del index + tamaño puntero))+1.** Esto es debido a que nuestro sistema gestor utiliza índices en árbol B.

8.5.2.2. Proceso práctico.

Se procederá a llevar a la práctica las ideas teóricas expuestas en la sección anterior.

Como primer paso, se procede a generar los datos a partir de una única consulta. Según la deducción teórica el alto grado de accesos a disco debería hacer inviable la consulta. Con la consulta siguiente:

```
insert into translator_a_to_b (id, name, id2)
SELECT   P.`id`,pes.id   FROM   idiomaA.Page   P,   idiomaB.Page   pes,
idiomaA.langlinks pl
where pl.ll_from=pes.pageid and
pl.ll_title = P.name and
pl.ll_lang='en';
```

Suponiendo los siguientes datos:

N_A	384.497 filas
N_B	1.660.067 filas
N_{AL}	52.258 filas
T_A	3630 bytes
T_B	3557 bytes
T_{AL}	32 Bytes
BL	8192 bytes

Tabla 3 - Datos de tablas

La consulta global utilizando un índice tendría que generar, según lo estudiado en la sección anterior, un número aproximado de accesos a disco de **`([F8] + [F4])`**, que con los datos con los que trabajamos ascienden a, **85.589.832** accesos a disco, suponiendo que el tiempo medio de acceso a disco es de **8,5 milisegundos**, supone un tiempo aproximado de **84 días**, contando como despreciable el tiempo de localización, trabajo en memoria y sin tener en cuenta, la escritura en disco de los campos coincidentes con la consulta que se deberán incluir en la tabla resultante.

Como bien predecía el resultado teórico, la consulta no la puede llevar a cabo nuestro sistema.

La segunda solución propuesta, que versa en dividir la consulta en dos subconsultas independientes:

```
insert into temp_lang (id,ll_title)
SELECT P.`id`, l.ll_title FROM IdiomaA.Page P, idiomaA.langlinks l
WHERE P.`pageId`=l.ll_from and
l.ll_lang = 'A';
```

Una primera consulta para generar la tabla temporal `temp._lang`, obtenemos directamente el `id` y filtramos por nuestro idioma deseado.

Con los datos expresado anteriormente en la tabla y los valores teóricos calculados [F4], el resultado de ejecutar esta consulta es de **192.454** accesos a disco, que suponen aproximadamente **27 minutos** teóricos para realizar la consulta, que coinciden con el resultado obtenido en la práctica.

El segundo paso de esta segunda solución propuesta consiste en rellenar la tabla final `translator_a_to_b` a partir de la tabla temporal y la tabla `Page` del idioma B:

```
insert into translator_a_to_b (id,ll_title,id2)
select l.id, l.ll_title, t.id
from temp_lang l, idiomaB.Page t
where l.ll_title= t.name;
```

Según la expresión obtenida teóricamente [F8], suponen **19.801.750** accesos a disco, que es un tiempo aproximado de 19 días. Coincidiendo con el resultado teórico, el experimento práctico tampoco era viable para llevarlo a cabo.

Para proceder con el tercer proceso, generamos un índice para el campo `name` de la tabla `page` del idioma B, un proceso bastante pesado que en dos horas se ha realizado.

Una vez generado el índice procedemos a repetir la consulta anterior

```
insert into translator_a_to_b (id,ll_title,id2)
select l.id, l.ll_title, t.id
from temp_lang l, idiomaB.Page t
where l.ll_title= t.name;
```

Como se ha explicado en la parte teórica, ahora la consulta la realiza a través de una estrategia de índice y requiere un número de accesos aproximado de [F9], que corresponden a **26254** accesos, un tiempo total de **3,7 minutos** que coinciden con el tiempo que ha tardado la máquina.

8.5.3. Conclusión

Tras los resultados obtenidos en esta sección, vemos que un estudio previo a la generación de una base de datos, resulta imprescindible. Para obtener un mismo producto, se ha pasado de consultas que consumían tiempos grandes y se preveían inviables a consultas viables y efectivas que se ejecutan en periodos de tiempo considerables, gracias todo ello a un minucioso estudio sobre la optimización de consultas.

8.6. Estudio de la estructura de datos.

8.6.1. Introducción

Una vez generados los datos, se debe implementar una arquitectura óptima de trabajo. Procederemos estudiando los problemas más simples que tenemos que resolver para ir aumentando poco a poco la complejidad y llegar a los problemas más complejos, toda vez para abarcar una solución común para todos los casos posibles.

Como principio general y después de un exhaustivo estudio de los motores de almacenamiento de MySQL (véase arquitectura de MySQL en estado de la cuestión), el sistema de tablas elegido para nuestro propósito es el de MyISAM. Una de las características técnicas de MyISAM, es no poder establecer claves ajenas, con lo que nuestro sistema de tablas quedará bastante simplificado.

Los problemas a solucionar usando Wikipedia, los evaluamos a continuación:

1) Traducir una palabra desde un idioma a otro, sin necesidad de obtener el texto de la entrada sobre la misma.
2) Buscar una traducción desde un idioma dado hacia cualquier otro idioma.
3) Traducir una palabra desde un idioma a otro y poder encontrar la traducción desde este segundo idioma hacia el primero dado.
4) Poder obtener el texto de la entrada entre traducciones de dos idiomas dados.
5) Poder obtener el texto de la entrada entre traducciones de múltiples idiomas dados.

Una vez realizadas las soluciones procederemos a realizar unas pruebas de funcionamiento. Las pruebas a realizar serán consultas de traducción directas y consultas de traducción con información semántica. El procedimiento consistirá en realizar aproximadamente 10 pruebas y mostrar la media de tiempo de todas ellas.

Para no contaminar el resultado de las pruebas, resulta de vital importancia indicar a MySQL que no guarde la consulta en caché, de otro modo, el sistema de pruebas sería contaminado, toda vez que el sistema no accedería a la base de datos, sino que obtendría los resultados de la memoria caché. Para realizar un sistema de pruebas adecuado, solicitaremos siempre la traducción de los mismos campos, que se muestran en la siguiente tabla:

Sin información Semántica	Con Información Semántica
Granularity	Isoflavone
Pizzaro	ISA
Brasero	Isobar
Syndication	Euphrosyne
Cadaveria	Dieting
Phytosterol	Coristanco
Xi_Persei	Tai_Chi
Badiraguato	Code
London	Topical
Capsule	Stimulus

Tabla 4 - Listado de palabras a traducir

8.6.2. Análisis 1:

El problema a resolver en este apartado es la traducción unidireccional desde el idioma A al idioma B utilizando la Wikipedia. No será necesaria la recuperación de información de la palabra para este análisis.

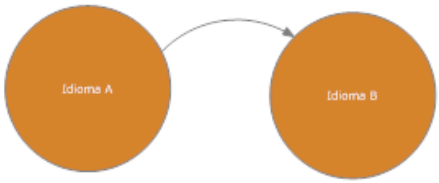

Figura 13 - Análisis 1

8.6.2.1. Solución aportada:

Para este tipo de traducción, hemos añadido a nuestro sistema una nueva tabla que contiene el `id` de la página y un campo de texto que representa su correspondiente traducción con el idioma B. Para acceder a la traducción debemos seleccionar el `id` de la tabla `page` y combinarlo con el id de la tabla `langB_outlinks`, a partir de ahí seleccionamos el campo titulo para obtener la traducción necesaria.

8.6.2.2. Arquitectura de la Base de Datos:

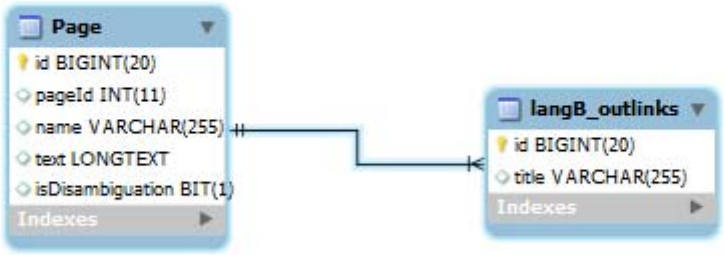

Figura 14 - Esquema relacional Análisis 1

8.6.2.3. Pruebas sobre la solución.

Con esta solución implementada, vamos a realizar las pruebas indicadas.

El primer paso es generar la tabla expuesta en el esquema e introducir los datos dentro de la misma, una vez realizado procedemos a realizar consultas:

```
SELECT sql_no_cache E.`ll_title` FROM langB_outlinks E, Page p
where
E.id=p.id and
p.name = 'Syndication';
```

los tiempos de respuesta para las pruebas son los siguientes:

Consulta	Tiempo
Granularity	0,0017s
Pizzaro	0,0017s
Brasero	0,0018s
Syndication	0,0018s
Cadaveria	0,0018s
Phytosterol	0,0018s
Xi_Persei	0,0025s
Badiraguato	0,0022s
London	0,0017s
Capsule	0,0017s

Tabla 5 - Resultados análisis 1

El tiempo de acceso medio para esta solución es de: **0,00187s.**

8.6.3. Análisis 2

El problema a resolver trata en realizar una traducción unidireccional desde un idioma A hacia cualquier otro idioma q .

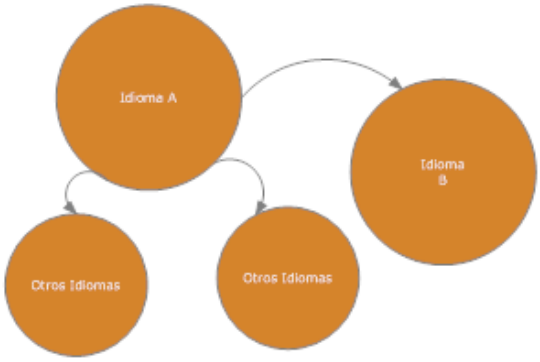

Figura 15 - Análisis 2

8.6.3.1. Solución Aportada:

Para este tipo de traducción, hemos añadido a nuestro sistema una nueva tabla que contiene el `id` de la página, un campo de texto que representa su correspondiente traducción así como un campo de texto que nos da la información del respectivo idioma con el que estamos trabajando. Al igual que en el análisis 1, tan solo deberíamos acceder correctamente a nuestra respectiva tabla y obtener la información deseada combinando los campos `id` de la tabla `page` y el campo `id` de la tabla `lang_outlinks`, filtrar por el idioma deseado y obtener el campo título.

8.6.3.2. Arquitectura de la BBDD:

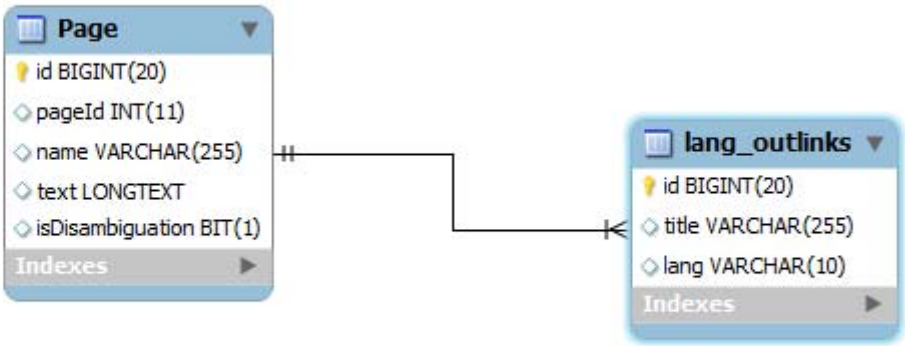

Figura 16 - - Esquema relacional Análisis 2

8.6.3.3. Pruebas sobre la solución.

Para realizar esta solución, hemos retocado la tabla del análisis uno y le hemos añadido el campo y los datos pertinentes a otros idiomas.

La consulta realiza ha sido muy parecida a la del análisis 1, salvo que se ha añadido un filtro por idioma:

```
SELECT sql_no_cache E.`ll_title` FROM langB_outlinks E, Page p
where
E.id=p.id and
p.name = 'Syndication' and E.lang='ES';
```

Los tiempos obtenidos son los siguientes:

Consulta	**Tiempo**
Granularity	0,0018s
Pizzaro	0,0018s
Brasero	0,0018s
Syndication	0,0018s
Cadaveria	0,0018s
Phytosterol	0,0019s
Xi_Persei	0,0025s
Badiraguato	0,0023s
London	0,0019s
Capsule	0,0019s

Tabla 6 - Resultados análisi 2

La media de tiempo es de : **0,0019s**

8.6.4. Análisis 3

El problema a resolver trata en realizar una traducción bidireccional desde un idioma A hacia un idioma B y viceversa.

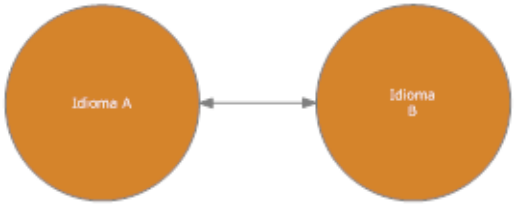

Figura 17 - Análisis 3

8.6.4.1. Solución Aportada:

La simetría de las arquitecturas explicadas en el análisis 1, hace factible y conveniente el uso de esa solución para este segundo problema.

Para acceder a la respectiva traducción tan solo deberemos acceder a la base de datos del idioma origen y realizar la misma combinación de tablas que hemos realizado en el análisis 1. Si por otro lado queremos que la traducción se produzca de manera inversa tan solo deberemos cambiar el acceso a la base de datos del idioma origen.

8.6.4.2. Arquitectura de la BBDD:

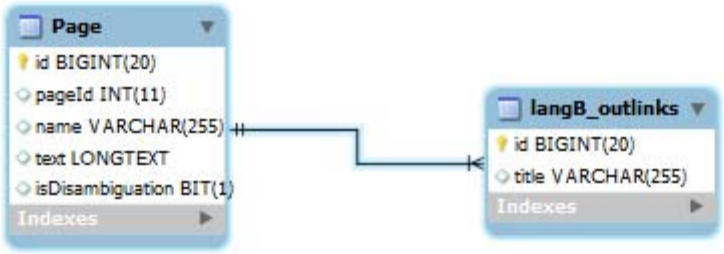

Figura 18 - - Esquema relacional Análisis 3

8.6.4.3. Pruebas sobre la solución.

Para este análisis, contamos con la solución del apartado uno, ya que la arquitectura es la misma.

8.6.5. Análisis 4

El problema a resolver trata en realizar una traducción unidireccional y obtener la información de dicha traducción, desde un idioma A hacia un idioma B.

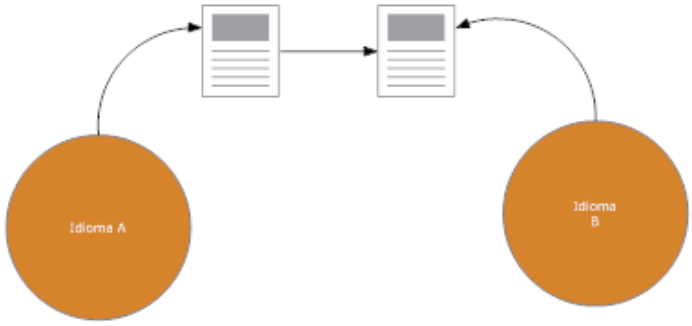

Figura 19 - Análisis 4

8.6.5.1. Solución Aportada:

La solución aportada para este caso consistiría en añadir una tabla llamada `Langb_outlinks`, que contendrá la información `id`, identificador de la página de la Wikipedia actual y `pageId` que contiene el campo `pageId` de la Wikipedia del idioma B. Para efectuar la traducción tan solo deberemos combinar los campos `id` de la tabla `Page` deseada y obtener el `pageId` de la tabla `langB_outlinks`. A partir de ahí accedemos al `pageId` de la base de datos de e otro idioma y obtenemos el texto de la entrada deseada.

8.6.5.2. Arquitectura de la BBDD:

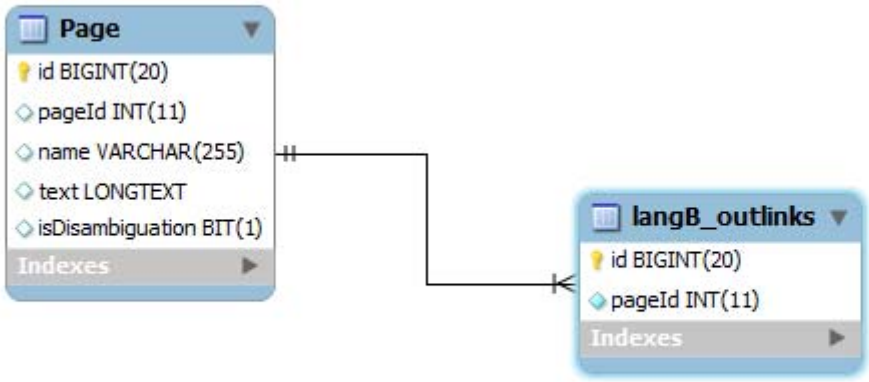

Figura 20 - - Esquema relacional Análisis 4

8.6.5.3. Pruebas sobre la solución.

Para esta arquitectura hemos retocado nuestra tabla y le hemos añadido un campo que contiene el `id` de la página destino.

Ahora en las pruebas, es necesario acceder a dos esquemas distintos, el esquema del idioma A donde consultaremos el nombre a traducir y obtendremos el `id` de la palabra del idioma B.

La consulta generada es de este estilo:

```
SELECT sql_no_cache  ES.`text` FROM langBoutlinks E, idiomaB.Page ES,
Page p where E.id=p.id and p.name = 'Capsule' and ES.id=E.pageID;
```

Los tiempos obtenidos para estas pruebas han sido los siguientes:

Consulta	**Tiempo**
Isoflavone	0,0026s
ISA	0,0030s
Isobar	0,0019s
Euphrosyne	0,0018s
Dieting	0,0019s
Coristanco	0,0019s
Tai_Chi	0,0026s
Code	0,0019s
Topical	0,0018s
Stimulus	0,0019s

Tabla 7 - Resultados análisis 4

El tiempo medio de acceso es de: **0,20s**

8.6.6. Análisis 5

El problema a resolver trata en realizar una traducción bidireccional y obtener la información de dicha traducción, desde un idioma A hacia un idioma B y viceversa.

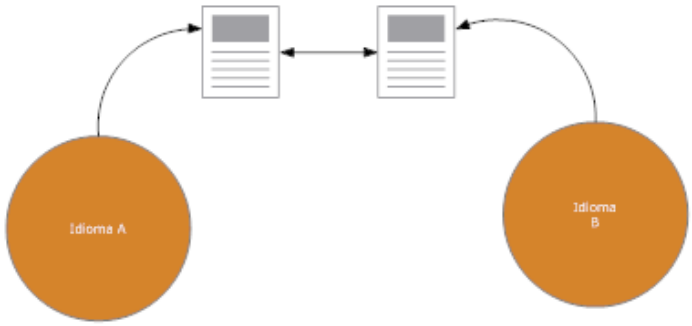

Figura 21 - Análisis 5

8.6.6.1. Solución Aportada:

Ante este problema nos surgen dos distintas soluciones.

La solución aportada en el apartado 4 es completamente válida, ya que podríamos contener la información dentro del esquema particular de cada Wikipedia y acceder igual que en el apartado anterior. (ver analisis 4).

La segunda solución aportada consiste en mantener los esquemas originales de la estructura de la base de datos optimizada y ampliar el sistema con la creación de un esquema intermedio que nos permita interrelacionar ambos.

Figura 22 - Solución Análisis 5

8.6.6.2. Arquitectura de la BBDD:

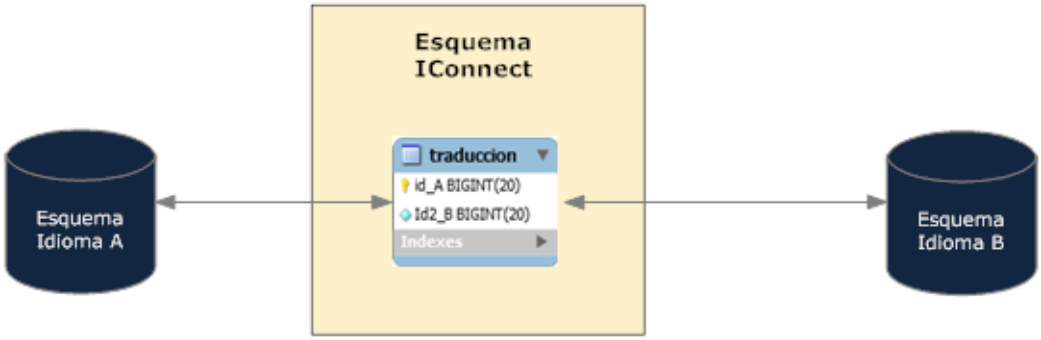

Figura 23 - - Esquema relacional Análisis 5

8.6.6.3. Pruebas sobre la solución

Hemos realizado nuevas pruebas para la segunda solución propuesta, ya que la primera solución propuesta ya se realizó pruebas en la sección anterior.

La consulta prototipo para estas pruebas es del estilo:

```
SELECT sql_no_cache P.`name` FROM idiomaB.Page P, idiomaA.Page EP,
interlenguaje.traduccion ET
where
P.id=ET.id1 and
ET.id2=EP.id
and EP.name='[Palabra]';
```

La siguiente tabla muestra los resultados obtenidos:

Consulta	Tiempo
Isoflavone	0,0033s
ISA	0,0034s
Isobar	0,0026s
Euphrosyne	0,0028s
Dieting	0,0028s
Coristanco	0,0027s
Tai_Chi	0,0026s
Code	0,0033s
Topical	0,0030s
Stimulus	0,0029s

Tabla 8 - Resultados análisis 5

El tiempo medio de acceso es: **29,4 s**

8.6.7. Análisis 6

El problema a resolver trata en realizar una traducción bidireccional y obtener la información de dicha traducción, desde un idioma A hacia otro idioma cualquiera.

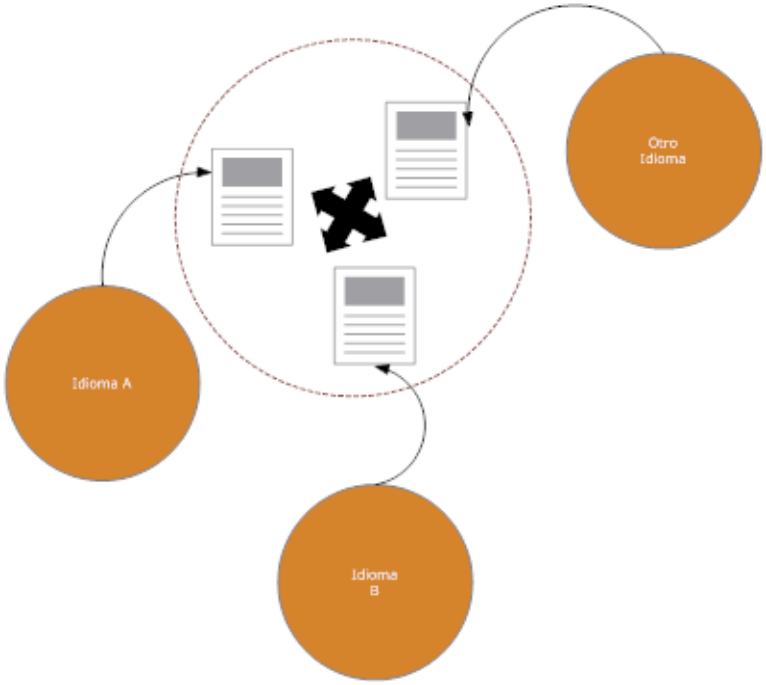

Figura 24 - Análisis 6

8.6.7.1. Solución Aportada:

Ante este problema solo podemos ver viable una solución, que consiste en generar una base de datos independiente al resto y asociar en ella el contenido de las distintas bases de datos, trabajar de otra manera generaría recurrencia de datos y una enorme complejidad a la hora de actualizar.

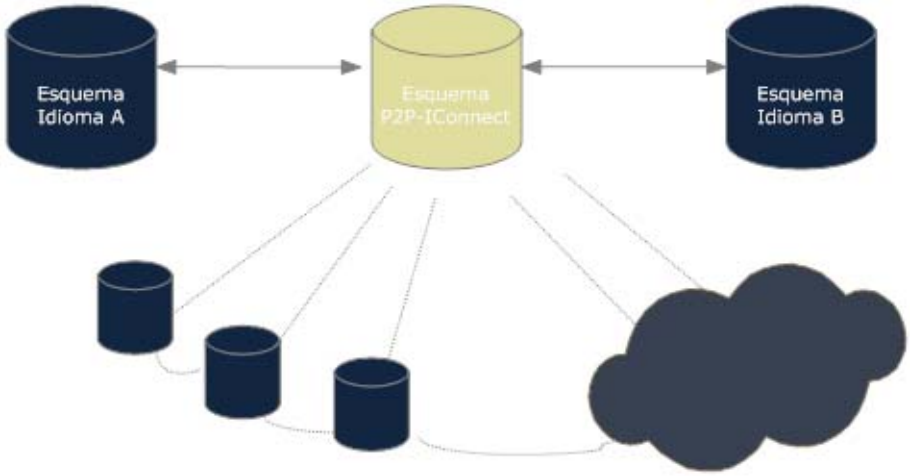

Figura 25 - Solución Análisis 6

8.6.7.2. Arquitectura de la BBDD.

Para este sistema podemos encontrar dos arquitecturas distintas, que podrían ser funcionales para nuestro sistema:

Esta arquitectura representa el modelo relacional más estricto y además nos permite versatilidad de una manera fácil a la hora de introducir distintas Wikipedia. Sin embargo, el introducir todas las relaciones dentro de la tabla `translate` podría significar demasiado tiempo de ejecución a la hora de realizar una consulta.

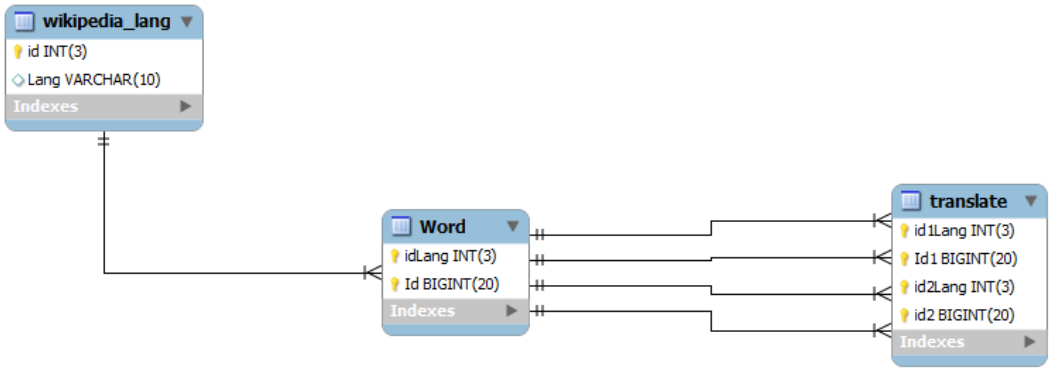

Figura 26 - - Esquema relacional Análisis 6

La segunda solución aportada parece ajustarse más al sistema que nosotros necesitamos, aunque se aparta sensiblemente de los estándares de las bases de datos relacionales. La solución básicamente consiste en la creación de una tabla plana que por cada fila contiene las distintas traducciones a cada idioma. Cada vez que nosotros quisiéramos añadir un idioma a nuestro sistema deberíamos añadir un campo nuevo a nuestra tabla.

Figura 27 - Esquema relacional 2 Análisis 6

8.6.7.3. Pruebas sobre la solución

Para este sistema las soluciones son las mismas que las expuestas en el análisis 4 y 5, por ello nos sirven los mismos resultados.

Por otra parte, la primera solución propuesta en este apartado, supone un sistema relacional bastante complejo para efectuarlo con las tablas MyISAM, por ello no resulta conveniente su implantación.

8.6.8. Conclusiones

Ante los resultados obtenidos en los anteriores análisis, realizaremos una tabla comparativa. Expondremos las soluciones que aporta cada una de ella a los distintos problemas en caso de que aporte solución.

Se procede a esquematizar los problemas del siguiente modo. A->B traducción de un idioma A a un idioma B. A <-> B traducción bidireccional entre idiomas. A ->[B] traducción de A a B obteniendo información semántica. [A] <->[B] Traducción de un idioma A a B obteniendo información semántica en ambas direcciones.

Solución	A->B	A<->B	A->[B]	[A]<->[B]
1	0,0018s	0,0018s	No	No
2	0,0019s	0,0019s	No	No
3	0,0018s	0,0018s	0,0020	0,0020
4	0,0029s	0,0029s	0,0029s	0,0029s

Tabla 9 - Resultados globales

Como podemos comprobar, la solución primera y segunda son incompletas y no pueden darnos información semántica de las palabras, con lo que las dejamos descartadas. Estudiando las soluciones 3 y 4 que son válidas para nuestro sistema, nos damos cuenta de que los tiempos de la solución número cuatro son demasiado elevados en comparación con los de la solución 3, esto es debido a que por cada consulta que debe realizar, la solución 4 tiene que acceder a tres esquemas distintos, con lo que el tiempo de acceso aumenta notablemente. Por otro lado, la solución 3 aporta unos tiempos inferiores, sin embargo, exige la duplicación de datos..

Por todo lo expuesto anteriormente y estudiando el objetivo de nuestro estudio, debido principalmente que para el trabajo con PLN es el tiempo de respuesta lo que más penaliza el funcionamiento sin importar la duplicidad de datos, ya que el tamaño en disco no es problema para nuestra arquitectura, elegimos como solución más efectiva la solución número 3.

8.7. Arquitectura del Sistema

8.7.1. Introducción

Se procederá a explicar las características técnicas y prácticas del nuevo sistema.

8.7.2. Modelo de Relaciones entre idiomas.

El modelo de relaciones entre idiomas, resulta bastante simplificado. Cada sistema de traducción contendrá un par de tablas que contiende la información a traducir, una tabla con la información para la traducción desde A a B en el esquema del idioma A y una tabla con la información para la traducción desde B a A en el esquema del idioma B.

Por ello, se realizará una implementación nueva del sistema por cada traducción que deseamos utilizar, es decir, una implementación nueva por cada par de idiomas. Por poner un ejemplo si queremos realizar traducciones entre tres idiomas diferentes A, B y C, se deberá implementar el sistema de traducción entre A y B, entre B y C y entre C y A. En total tres sistemas para poder tener un sistema de traducción completo.

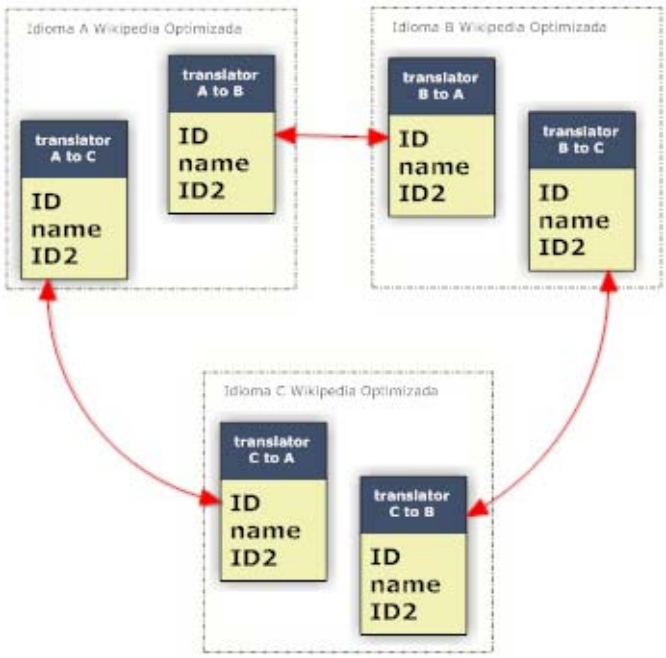

Figura 28- Ejemplo de sistema para 3 idiomas

Según la naturaleza del sistema, la tabla de traducción del idioma A y la tabla de traducción del idioma B, se puede decir que contienen información inversa, a pesar de que esta afirmación no es cierta del todo, ya que existe un campo nombre que no se duplica, pero se podría afirmar que existe cierta duplicidad de datos en los distintos esquemas. Brevemente podemos afirmar que es una penalización a admitir para poder trabajar con la velocidad de recuperación adecuada para PLN, debido a la indexación de campos.

8.7.3. Modelo Estructural.

El modelo estructural del sistema es muy parecido al original de JWPL. Es decir, se ha trabajado sobre el sistema optimizado de Wikipedia y le hemos añadido, un **índice en árbol B para el campo** `name` de la tabla `Page`, **una tabla de traducción por cada idioma a traducir**.

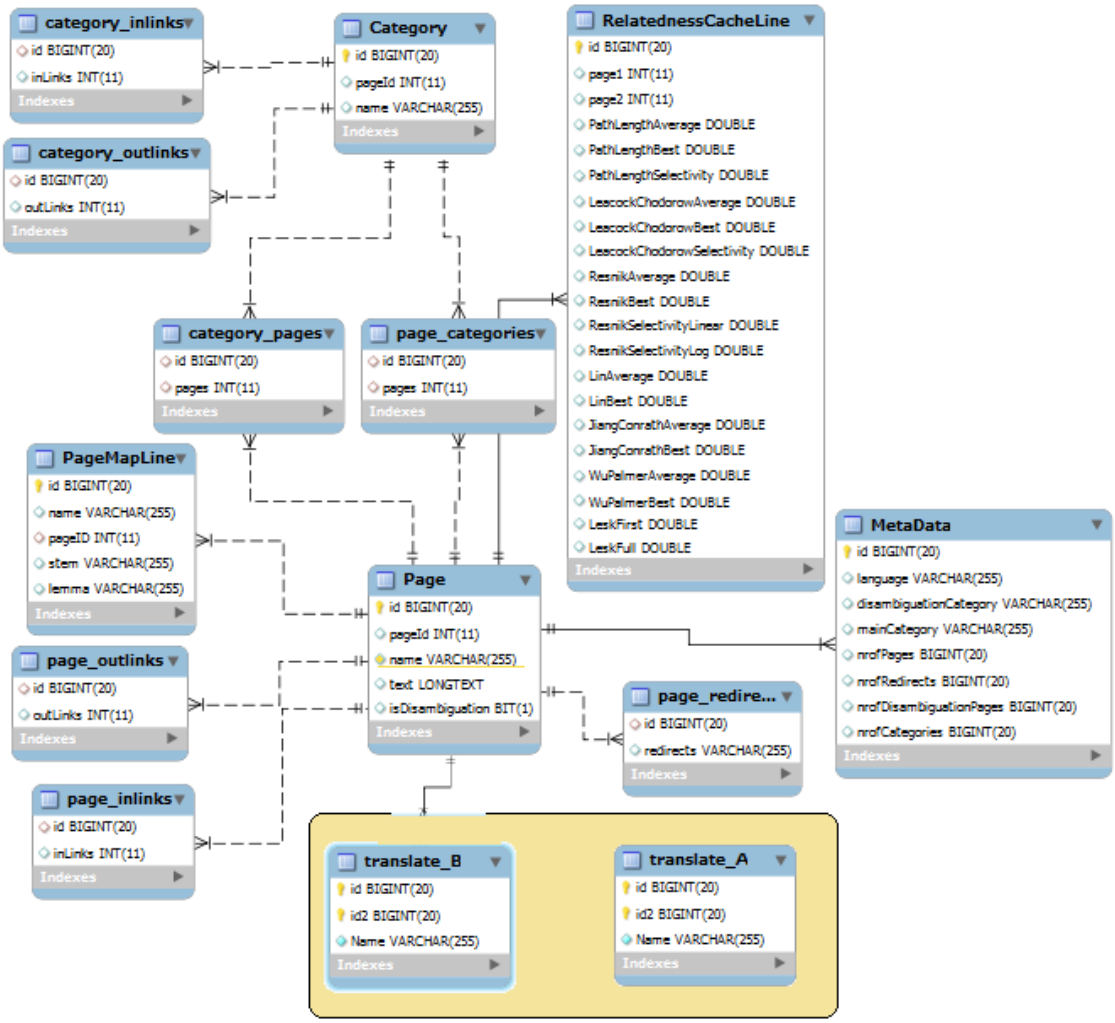

Figura 29 - Estructura final de Transwiki

MySQL realiza la indexación del campo `name` mediante un árbol B, esa indexación supone que los accesos a disco en busca de un campo disminuyan a $\log(N_A)/\log(BL/(3 \times 2)/(\text{longitud del index} + \text{tamaño puntero}))+1$, para casos en que haya muchos campos y 1 acceso para índices pequeños. En nuestro ejemplo, gracias a esta indexación hacemos factible al sistema para trabajar con PLN.

Además de ello, el sistema dispondrá de tantas tablas `translate` como sistemas de traducciones tenga implementados.

La tabla contiene un campo identificador de página del idioma propio, un campo `name` que representa el string de traducción del idioma foráneo. Este campo está estrictamente pensado para las traducciones que no requieran de información semántica de la palabra, ya que de este modo ahorraremos tiempo en no tener que acceder al esquema del idioma foráneo a recuperarlo. El sistema de traducción

directa se puede preveer como el sistema de traducción que más se utilizará para trabajos con PLN, por ello, con este campo ahorramos por cada consulta una media de 0,0002s, tiempo nada despreciable cuando las consultas se multiplican. Además de esto, deberemos tener en cuenta que la memoria caché de consultas que implementa el SGBD MySQL, disminuirá notablemente el tiempo de respuesta para consultas directas.

La utilización de memoria caché de consultas, es un factor clave a la hora de trabajar con traducciones toda vez, que por naturaleza de los lenguajes, es alto el grado de frecuencia léxica que intervienen en un texto, por ejemplo, las siguientes palabras son las más repetidas en los textos Castellanos:

De	691.341	6,68%
La	445107	4,30%
Que	319892	3,09%

Tabla 10 - Frecuencia léxica de Corpus-Vox Bibliograf, *Manuel Alvar Ezquerra*.

Estas repeticiones se incrementan contra más grande sea el tamaño del texto. Así, según las informaciones de *Patterson y Urrutibéheity* nuestras 100 palabras más frecuentes constituyen más del 30% del material léxico de cualquier texto; con las 1000 se alcanza el 50% de todo texto, y con las 5000 más abundantes se sobrepasa el 90%.

Por todo ello, gracias a la memoria caché, nuestro sistema está preparado para ser más efectivo contra más grande sea el tamaño del texto a traducir, ya que disminuirán notablemente el número de accesos a disco.

En contraposición con lo explicado, la pragmática léxica hace que tengamos que acudir a recoger el texto de la entrada de la palabra. Para ello nuestro sistema, tendrá que buscar en dos esquemas diferentes, en los motores MyISAM de MySQL cada esquema se guarda en un fichero diferente, con lo que el sistema tendrá que abrir dos ficheros distintos para obtener los datos y la consulta se verá penalizada en tiempo. Por suerte, esta penalización se ve disminuida ya que al trabajar con campos indexados los accesos a disco son mínimos como ya hemos explicado anteriormente.

A pesar de todo ello, la recuperación del texto de la entrada de una palabra solo se ve afectada en un tiempo total de 0,002s y teniendo en cuenta que los casos de pragmática léxica suponen menos del 10% para textos grandes y poco ambiguo, la penalización en tiempo no será apreciable por el sistema.

8.8. Automatización del proceso

El sistema transwiki conlleva una serie de pasos de creación bastante complejos y difíciles de explicar, por ello, es necesario encapsular dicha información al usuario y facilitar el proceso creando un código de generación automático que a partir de los datos facilitados por el usuario nos deje el sistema dispuesto para su utilización.

A la hora de generar un código nos surge la idea de que lenguaje utilizar, con ello se proponen múltiples posibilidades. Tenemos la opción de utilizar un código de programación compilado como es Java, sin embargo, la utilización de dicho código puede dificultar al usuario final su utilización, ya que depende directamente de instalar los complementos Java necesarios. Por otro lado podríamos utilizar lenguaje PHP, cargando las sentencias sql, pero este lenguaje también necesita de su correspondiente intérprete y además disminuye notablemente el tiempo de ejecución de las tareas. Cualquier de estos lenguajes podría generar el código para nuestro usuario final, pero bien es cierto que complicaría notablemente su ejecución, por ello, se ha decidido a realizar el código en un script de shell de linux ya que al venir integrado directamente con los sistemas UNIX, no hace falta de complementos y se puede instalar directamente.

El script generado, creará traducciones para pares de idiomas, por ello deberemos ejecutar tantas veces el script como distintas traducciones entre idiomas se quiera crear.

8.8.1. Estructura del script.

El script generado para este sistema viene incluido en el anexo y se procederá a su explicación.

El script recibe 6 variables de entrada para su ejecución:

- Variable 1: Nombre de la Base de datos de la Wikipedia para el idioma A.
- Variable 2: Nombre de la Base de datos de la Wikipedia para el idioma B.
- Variable 3: Nombre de el DUMP de Wikipedia langlinks para el idioma A, es importante que se incorpore con la misma extensión que se descarga de la página Web, es decir, [NOMBRE].sql.gz
- Variable 4: Igual que la variable 3 pero para el idioma B.
- Variable 5: Es el langcode de Wikipedia para el idioma A, este Langcode correponde con el prefijo incorporado para las páginas web de cada país de Wikipedia. Para el castellano "es".
- Variable 6: Corresponde al langcode de Wikipedia del idioma B.

La primera parte del código el programa solicita los datos de conexión a la base de datos MySQL, para ello necesitamos introducir los datos de un usuario con permiso de administrador en ambas base de datos.

El programa procede a cargar los DUMPS de Wikipedia en la base de datos.

Una vez cargados los dumps de la Wikipedia el sistema genera un índice para la tabla `Page` la cual indexamos por el campo `name`.

Una vez generado ese índice procede a crear una tabla temporal llamada `lang_temp` compuesta por el campo `name` e `id` .

Posteriormente se procede a filtrar los datos cargados en `langlinks` por el idioma pertinente y se traducen por el `id` de la Wikipedia optimizada, se introducen en `lang_temp`.

Generamos un tabla translator_[langcode idioma destino] que contiene los campos `id`, `name`, `id2` que representa el id de la traducción.

Posteriormente realizamos un proceso de mezcla de datos, como ya se explicó, para compensar los datos entre las distintas tablas y así obtener el máximo de información posible.

Realizamos una limpieza del sistema, borrando las tablas temporales que no nos interesan.

8.8.2. Ejecución del Script.

A la hora de ejecutar el script es imprescindible realizarlo sobre una shell de linux, en este proyecto para comprobar la máxima compatibilidad del script con las distintas versiones de Linux, hemos realizado la importación sobre máquina ubuntu y sobre una máquina SUSE.

Para invocar correctamente el script deberemos ejecutar el siguiente comando:

```
Sh install.sh [variable1] [variable2] [variable3] [variable4]
[variable5] [variable6]
```

Correspondiendo las variables anteriores con lo explicado en la sección anterior.

Un ejemplo para los idiomas Inglés y Español, sería el siguiente:

```
sh install.sh wikipedia_es wikipedia_en eswiki-20092706-
langlinks.sql.gz enwiki-20092706-langlinks.sql.gz es en
```

8.9. Sistema final implantado.

Finalmente para la puesta en marcha de este proyecto se ha llevado a cabo un servidor con las características técnicas comentadas en las secciones anteriores.

Este servidor consta de la información de las Wikipedias Optimizadas para los idiomas; Español, Checo, Arabe, Francés e Inglés.

Para cada uno de estos idiomas se ha generado el correspondiente sistema de traducción hacia los otros restantes, la arquitectura establecida finalmente resulta la mostrada en la imagen.

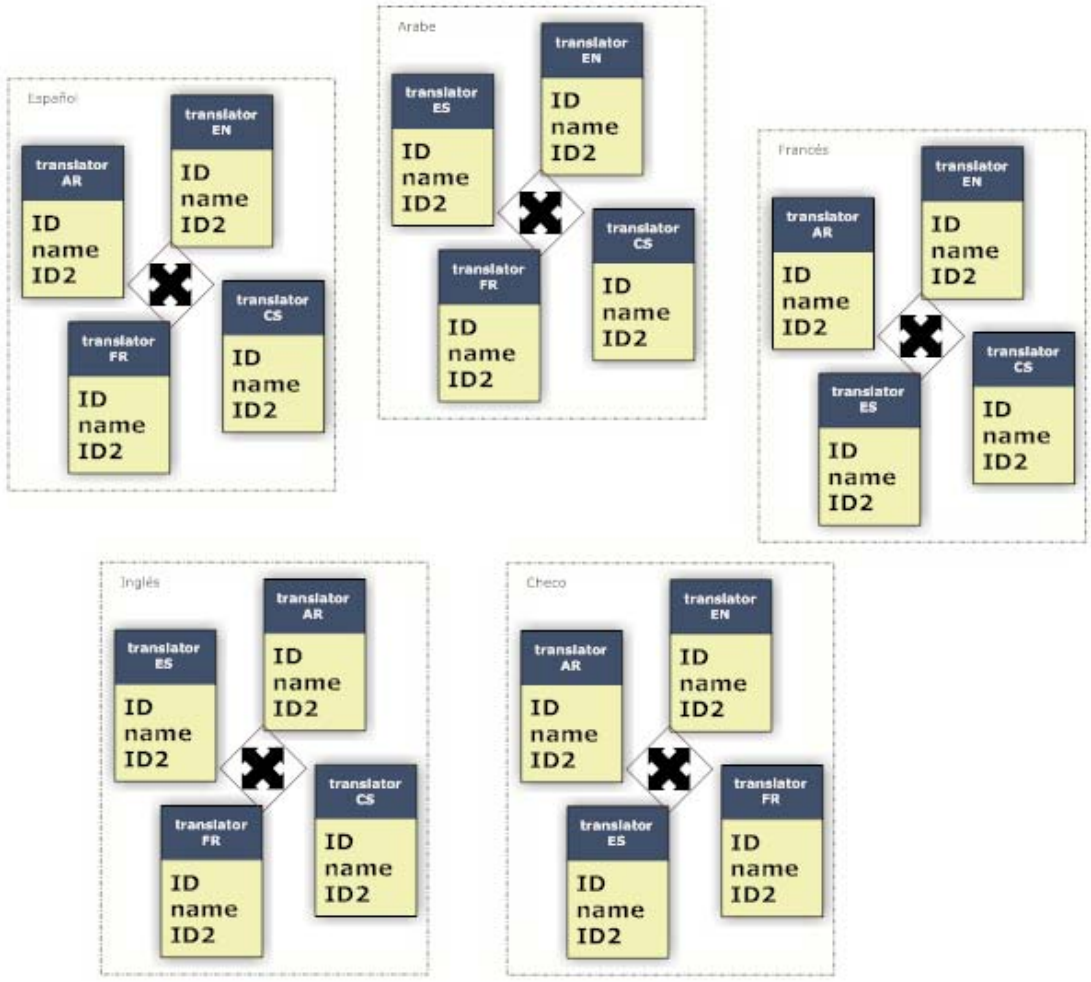

Figura 30- Sistema Final

9. Futuras Líneas de Investigación.

Este proyecto tiene la ventaja de ser una **excelente herramienta de apoyo** para muchos trabajos futuros. Hace años cuando comenzaban a estudiarse las bases de datos e implementarse para su uso en las grandes organizaciones, se contaba con estructuras de datos muy precisas y jerarquizadas, hoy en día se tiende a trabajar con mucha más información que antaño y prácticamente sin estructurar. Por ello requerimos de procesadores de lenguaje capaces de estructurar ó enfocar toda esa información hacia nuestros intereses. Todo ello hace que el trabajo realizado en este proyecto, esté ampliamente enfocado para su **adecuación en un futuro con procesadores del lenguaje natural** capaces de automatizarnos el proceso de estructurar la información entre distintos idiomas.

A grandes rasgos, lo expuesto anteriormente resulta un tratamiento futuro de este sistema, sin embargo, para facilitar ese paso futuro, se requiere de un paso previo que facilite notablemente el trabajo. Aprovechando que hemos integrado nuestro sistema conjunto a JWPL, resultaría muy interesante, lograr integrar una API en java con la API de JWPL, capaz de trabajar con esta y facilitar las tareas propias de la traducción, integrándolo así, con un fuerte lenguaje de programación. De esta manera los desarrollos futuros serán más fáciles, ya que tan solo tendrán que interactuar con la API de Java.

El trabajo de este sistema comentado en los párrafos anteriores, podrá ser utilizado en campos de la informática destinados a la inteligencia artificial, permitirá **tratar textos entre diversos idiomas e inclusive alinear frases entre distintos idiomas**. Indirectamente gracias a nuestro sistema, podremos estudiar la distancia que separa dos idiomas dados y predecir la ascendencia de palabras.

El trabajo más sencillo, y evidente, para una línea futura de desarrollo, es entre otros la creación de un **traductor enciclopédico**. De esta manera poder obtener la máxima información contenida entre las distintas Wikipedia.

Muchos son los campos donde se puede trabaja con este sistema, que además de ello se verá beneficiado por su bajo coste de implantación, debido a la gratuidad que suponen los datos de Wikipedia. Todo ello hace de este trabajo un camino de desarrollo futuro y una base para próximos avances en múltiples campos de la informática, en especial en el campo de la inteligencia artificial.

10. Conclusiones.

Dando por finalizado este trabajo, se puede concluir que se ha cumplido realizando todos los objetivos previstos para el mismo.

Primeramente se ha logrado estudiar y comprender la información incluida en la Wikipedia y lograr implementar con ella un sistema de traducción con unos tiempos de respuesta excelentes, que sin duda era el principal requisito para los fines de este sistema.

Por otro lado, se ha estudiado la infraestructura de JWPL y con ello se ha logrado integrar nuestro sistema de traducción con dicha infraestructura, de este modo resultará sencillo ampliar esta API en java para realizar sistema de traducciones y con ello facilitar y potenciar la utilización de este sistema de traducción.

Cumplidos todos estos requerimientos, se puede afirmar que disponemos de un sistema rápido, eficiente, en constante incremento debido a la naturaleza de las fuentes de donde procede la información y sobre todo gratuito y de fácil implementación. Además, es importante recalcar otra propiedad de este sistema que lo diferencia de los sistemas comunes de traducción, este sistema no está realizado por lingüistas especializados, sino que contiene una información más coloquial que no disponemos en los comunes traductores, por ello, resulta una fuerte herramienta para traducciones de términos no aceptados por las academias de las lenguas, nombres propios ó bien cualquier otro termino no incluidos en diccionarios pero que su uso se encuentre totalmente extendido.

Finalmente podemos afirmar que se ha generado un recurso básico para otros estudios y con ello concluir que puede ser la primera piedra para su utilización y explotación de la información que contiene.

11. Anexo.

11.1. Anexo 1. Proceso de instalación de un Servidor Media-Wiki.

Se procederá a explicar como realizar la instalación de un servidor funcional para alojar una propia Wikipedia.

11.1.1. Instalación de un servidor Ubuntu Linux.

La instalación de un servidor Ubuntu Linux, es bastante sencilla. Como primer paso debemos bajar una versión desde la página Web del proyecto Ubuntu.

Una vez descargada la versión e grabada correctamente en un CD, insertamos el CD y reiniciamos el equipo, como paso previo deberemos configurar como sistema de arranque prioritario en la BIOS el CD-ROM donde insertaremos Ubuntu.

Una vez arrancado, simplemente debemos seguir las instrucciones de instalación y procederemos a completar el sistema.

Si nuestra red cuenta con servidor DHCP, automáticamente al arrancar nuestro servidor se encontrará conectado a Internet, sino deberemos configurar los parámetros adecuados a nuestra red en el interfaz de red adecuada.

11.1.2. Instalación de Apache 2.0, php5 y lib.

La instalación de un servidor Apache 2.0 , es bastante sencilla si contamos con conexión a Internet. Para realizarla procederemos a utilizar el gestor de archivos apt-get que viene integrado con el sistema operativo Ubuntu.

```
sudo apt-get install apache2
```

El sistema automáticamente procederá con la descarga del paquete y su instalación.

Para que nuestro equipo apache responda a páginas Web dinámicas PHP, debemos instalarle el complemento PHP5 al sistema seguido del modulo libapache2-mod-php5 .

```
sudo apt-get install php5
sudo apt-get install libapache2-mod-php5
```

11.1.3. Instalación servidor MySQL y creación del esquema.

La instalación del servidor es igual de sencilla que las instalaciones anteriores

```
sudo apt-get install MySQL-server-5.0
```

Una vez instalado e introducida la contraseña de administración del sistema, deberemos proceder a configurar y crear un nuevo esquema que albergue los datos pertenecientes a la Wikipedia.

```
create schema Wikipedia_es
```

11.1.4. Recuperación de los DUMPS de Wikipedia.

Los DUMPS de Wikipedia, los podemos descargar de la siguiente página:
http://download.wikimedia.org/
En la sección de XML y SQL dumps podremos encontrar los datos de la Wikipedia en el idioma que deseamos y referente a las páginas que deseamos. Dependiendo de los datos, los dumps pueden estar en código SQL ó bien xml, para cada uno de estos casos deberemos utilizar una forma de recuperación diferente.

En el caso de que el dump se encuentre en formato estándar SQL, tan solo deberemos utilizar la utilidad mysqlimport, y automáticamente nos cargará el archivo SQL deseado. Si el archivo se encuentra en formato xml, tenemos varias aplicaciones capaces de importarlo a nuestra base de datos, una de ella es mwdump, que es una aplicación java de software libre.

11.1.5. Configuración de Media Wiki.

La configuración de media Wiki es un proceso bastante sencillo. Tan solo debemos descargar media Wiki de la página oficial y descomprimirlos en la carpeta de páginas Web configurada en apache. Ejecutamos la ruta local de la carpeta de media Wiki en el explorador Web y seguimos los pasos del sistema configurador. La ruta local a configurar en el explorador suele tener este formato:

```
http://localhost\mediawiki
```

Inmediatamente nos mostrará el configurador donde deberemos introducir datos tales como, nombre, usuario y contraseña de la base de datos de Wikipedia, nombre de la Web, caracteres por defecto, etc.

Un script interno de media Wiki realizará el archivo de configuración adecuado y nos pedirá que eliminemos la carpeta del configurador. Una vez eliminada la carpeta del configurador, tendremos media Wiki funcionalmente completo.

11.2. Anexo 2 . Traducción de datos.

11.2.1. Procedimiento.

El proceso definitivo para la creación de nuestro sistema consiste en los siguientes pasos:

```
alter table Page add Index ind_name (name);
```

Generamos un índice por el campo `name`. Es necesario ya que las traducciones siempre serán generadas a partir del campo `name`, y nos interesa que nuestro sistema encuentre lo más rápido posible. Una búsqueda normal sin índice para este campo dura aproximadamente un minuto, mientras que si indexamos, en menos de un segundo hemos devuelto los datos.

Cargamos el DUMP de nuestra Wikipedia donde vienen los enlaces a otros idiomas

```
gunzip < DUMPFILE.sql.gz | mysql –u USER -p --default-character-
set=utf8 DB_NAME
```

Es importante tener en cuenta que la tabla que genera este backup, está constituida por tres campos; el campo `ll_from`, es decir, el índice de la Wikipedia no estructurada, el campo `ll_name`, la traducción directa a un idioma dado, `ll_lang` es el campo que guarda el prefijo del idioma al que pertenece la palabra traducida.

Generamos una tabla temporal que nos guardará el `id` de la Wikipedia Optimizada y su palabra traducida. Esta tabla nos ayudará notablemente para agilizar pasos posteriores.

```
CREATE TABLE  `wikipedia_est_EN`.`lang_temp` (
  `name` varchar(255) NOT NULL,
  `id` bigint(20) unsigned NOT NULL,
  PRIMARY KEY  USING BTREE (`id`),
  KEY `indice_lang_temp` (`name`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

El siguiente paso es filtrar las palabras de los idiomas que nos interesa y cambiamos el `ll_from` que representa el índice de la antigua Wikipedia, por nuestro `id` de la Wikipedia Optimizada.

```
insert into lang_temp (id,name)
SELECT P.`id`, l.ll_title FROM Page P, langlinks l
WHERE l.ll_from=P.`pageId` and
l.ll_lang = 'es';
```

Esta consulta es bastante pesada, debido principalmente al gran número de campos que hay dentro de la tabla.

Eliminamos la tabla `langlinks` que ya no vamos a utilizar.

```
drop table langlinks;
```

Generamos la tabla final que albergará la información necesaria para nuestro sistema de traducción bidireccional.

```
CREATE TABLE  `wikipedia_est_EN`.`ES_translater` (
  `id` bigint(20) unsigned default NULL,
  `ll_title` varchar(255) character set latin1 default NULL,
  `id2` bigint(20) unsigned default NULL,
  KEY `pt_id` (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

Esta tabla contiene el id de la página a traducir, su correspondiente palabra traducción y el id de la página que representa la traducción, para el caso en que necesitásemos adquirir el texto de la entrada de una palabra.

```
insert into ES_translater (id,ll_title,id2)
select l.id, l.name, t.id
from lang_temp l, wikipedia_est_ES.Page t
where l.name= t.name;
```

Con esta consulta introducimos el `id`, el nombre y el `id` de la palabra traducida seleccionando todos los campos de ambas tablas bajo las cuales coinciden los títulos. Es importante tener en cuenta que para esta consulta es necesario acceder conjuntamente a dos esquemas distintos y esto aumenta notablemente el tiempo de ejecución.
Para dejar limpia nuestra base de datos, eliminamos la tabla temporal `lang_emp`.

```
drop table lang_temp ;
```

Una vez generadas las dos tablas, deberían de coincidir el número de campos de ambas tablas, sin embargo, debido a la naturaleza de Wikipedia, esto no sucede. Es decir, pongamos un ejemplo para aclararlo. Un usuario que introduce información para la palabra inglesa "house" es probable que introduzca su correspondiente página en la Wikipedia española "casa", sin embargo, puede proceder que este usuario no haya dado de alta el enlace de la Wikipedia española hacia la Wikipedia inglesa. Debido a este problema, tenemos descompensados el número de campos en ambas tablas y lo lógico será aprovechar al máximo la información que poseemos para poder traducir el máximo número de campos desde ambos lenguajes. Por ello debemos de buscar para una Wikipedia A las traducciones que no se encuentran en la Wikipedia B e introducirlas en B y hacer el mismo procedimiento con la Wikipedia A.

Para realizar este proceso hemos optado por realizar una consulta en SQL que nos mezcle los campos y nos rellene cada tabla con la información que falta, para nuestras pruebas realizamos la siguiente consulta para el idioma A:

```
insert into prueba_EN_traslater (id, ll_title, id2)
select t.id2, p.name, t.id from wikipedia_est_EN.prueba_ES_traslater
t, wikipedia_est_EN.Page p
```

```
where
p.id=t.id and
(t.id, t.id2) not in (select e.id2, e.id from prueba_EN_traslater e);
```

Y posteriormente la siguiente consulta para el idioma B,:

```
insert into prueba_ES_traslater (id, ll_title, id2)
select t.id2, p.name, t.id from wikipedia_est_ES.prueba_EN_traslater
t, wikipedia_est_ES.Page p
where
p.id=t.id and
(t.id, t.id2) not in (select e.id2, e.id from prueba_ES_traslater e);
```

Ahora tan solo nos queda comprobar que las dos tablas poseen el mismo número de campos:

```
SELECT count(p.`id`) FROM prueba_ES_traslater p;
```

```
63978
```

```
SELECT count(p.`id`) FROM prueba_EN_traslater p;
```

```
63978
```

Podemos finalizar el trabajo para esta traducción bidireccional.

11.2.2. Conclusión.

La mejor solución encontrada para el sistema de traducción bidireccional, consistiría en la generación de dos tablas por sistema bidireccional de traducción, es decir, para dos idiomas A y B se generaría una tabla en el esquema de A con los datos para la traducción con B y otra tabla en el esquema de B para traducir en A. Dentro de cada tabla guardaremos los siguientes campos, `Id` de la página origen, `Id` de la página traducción con el fin de acceder cuando sea necesario a el texto de la entrada de la palabra y un tercer campo que contiene el titulo de la página a traducir, es decir, la cadena de texto traducción a la página a traducir, ya que de esta manera obtendremos traducciones más rápidas en campos donde no exista ambigüedades y no haya que acceder a la información de la páginas.

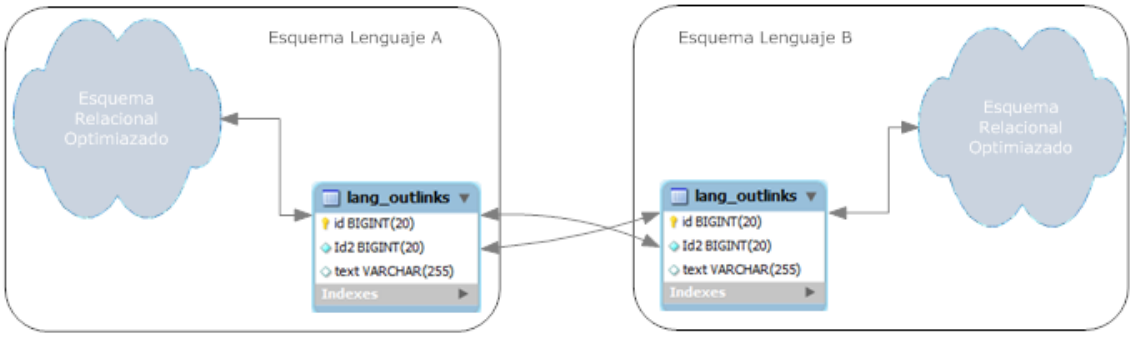

Figura 31- Arquitectura del esquema solución

11.3. Anexo 3 . Esquema de la Wikipedia.

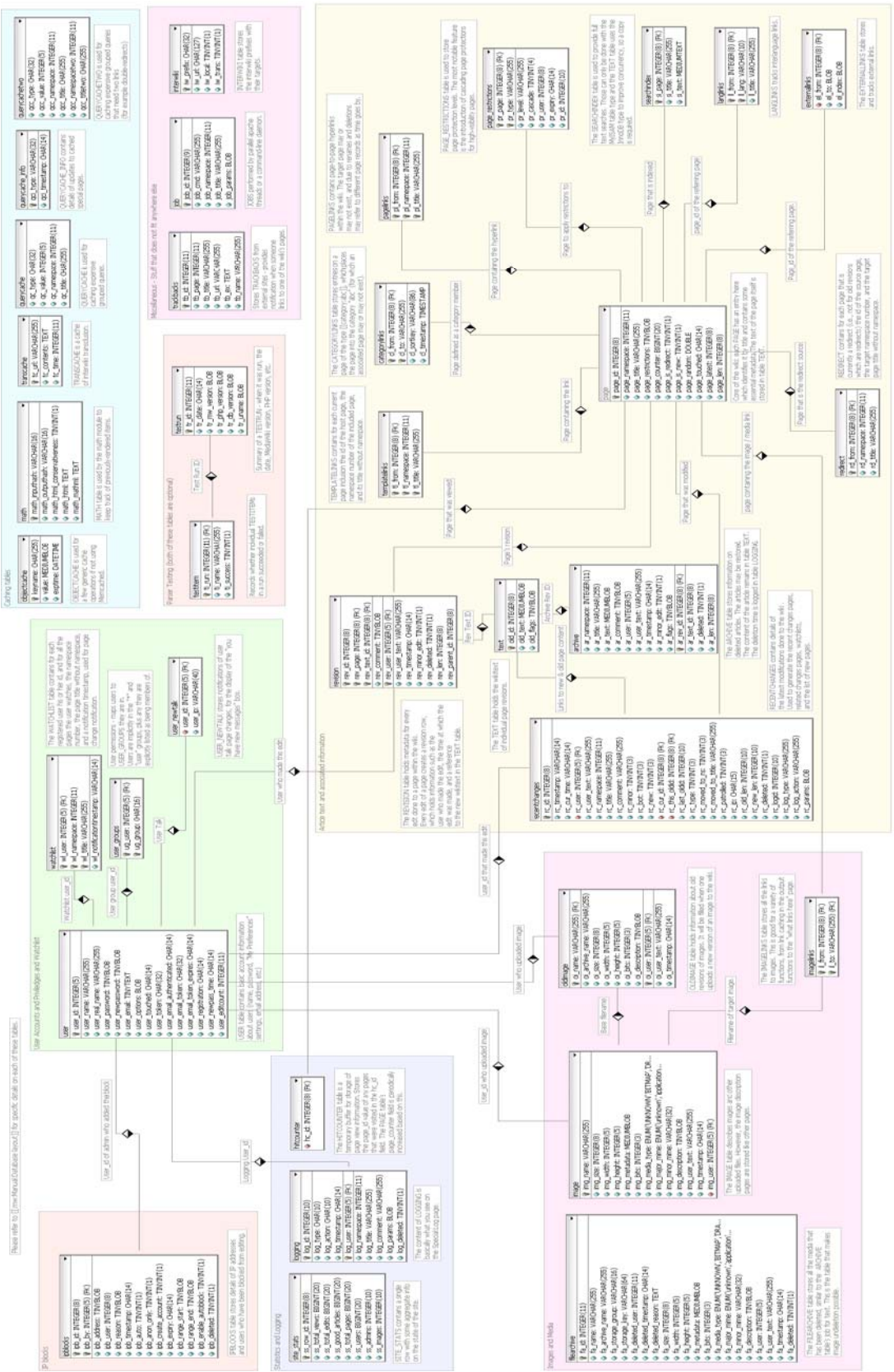

Figura 32 - Esquema de la wikipedia

11.4. Anexo 4. Ejemplo de Página de la Wikipedia.

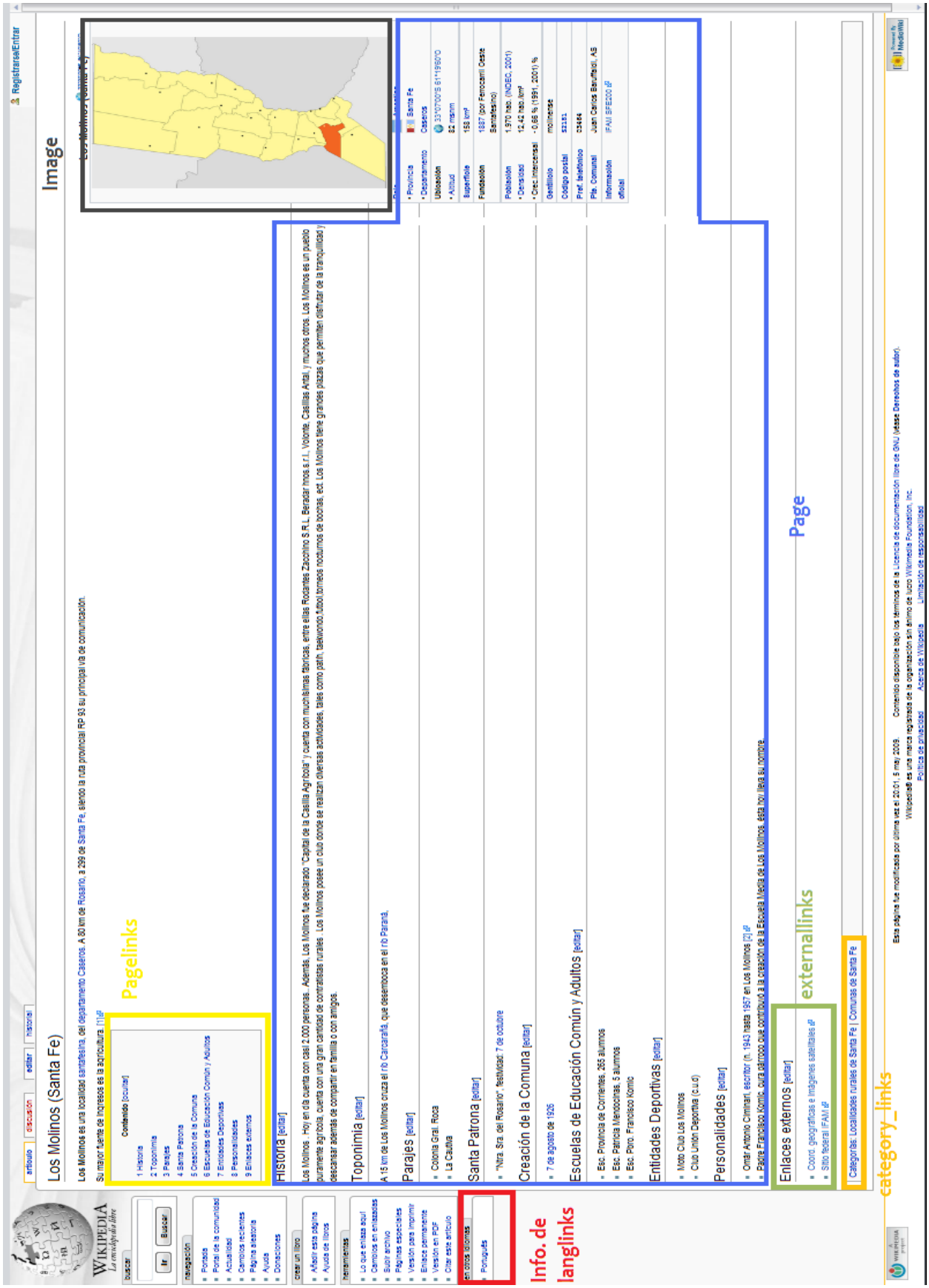

Figura 33 - Ejemplo de Web de wikipedia.

11.5. Anexo 5 . Script de instalación.

```
#validating input parameters
if [ $# -ne 6 ];
      then
      echo "Incorrect number of parameters"
      exit 1
fi
#correct number of parameters
#requesting data connection
echo "Intro MySQL USER:"
read usr
echo  "Intro MySQL host name:"
read hst
stty -echo
echo "Intro MySQL password:"
read pass
clear
stty echo
###Loading DUMP for language 1
echo "Loading DUMP- langlinks for $1, please be patient, it may take a
while"
mysql -h $hst -u $usr -p$pass $1 -e "alter database $1 charset = utf8"
gunzip  <  $3  |  mysql  -h  $hst  -u  $usr  -p$pass  --default-character-
set=utf8 $1
if [ $? -ne 0 ];
      then
            echo "error loading DUMP for $s1 "
            exit 1
fi
echo "Loading DUMP- langlinks for $2, please be patient, it may take a
while"
###Loading DUMP for language 2
mysql -h $hst -u $usr -p$pass $2 -e "alter database $2 charset = utf8"
gunzip  <  $4  |  mysql  -h  $hst  -u  $usr  -p$pass  --default-character-
set=utf8 $2
if [ $? -ne 0 ];
      then
            mysql -h $hst -u $usr -p$pass $1 -e "drop table langlinks;"
            echo "error loading DUMP for $s2"
            exit 1
fi
####generating index language 1
echo  "generating  index  for  $1.Page.name,  please  be  patient,  it  may
take a while /n this process may fail in seconds instalations"
mysql  -h  $hst  -u  $usr  -p$pass  $1  -e  "alter  table  Page  add  index
page_name (name)"
######generating index language 2
echo  "generating  index  for  $2.Page.name,  please  be  patient,  it  may
take a while /n this process may fail in seconds instalations"
mysql  -h  $hst  -u  $usr  -p$pass  $2  -e  "alter  table  Page  add  index
page_name (name)"
#creating temporal table lang_temp for language1
echo "creating temporal table for $1.lang_temp, please be patient, it
may take a while"
mysql -h $hst -u $usr -p$pass $1 -e "CREATE TABLE  $1.lang_temp ( name
varchar(255)  NOT  NULL,  id  bigint(20)  unsigned  NOT  NULL,  PRIMARY  KEY
USING  BTREE  (id),  KEY  indice_lang_temp  (name))  ENGINE=MyISAM  DEFAULT
CHARSET=utf8;" > lang_temp.log
```

```
if [ $? -ne 0 ];
      then
            mysql -h $hst -u $usr -p$pass $1 -e "drop table langlinks;"
            mysql -h $hst -u $usr -p$pass $2 -e "drop table langlinks;"
            echo "error creating table for $s1.lang_temp \n Read
lang_temp.log"
            exit 1
fi
rm lang_temp.log
#creating temporal table lang_temp for language2
echo "creating temporal table for $2.lang_temp, please be patient, it
may take a while"
mysql -h $hst -u $usr -p$pass $2 -e "CREATE TABLE  $2.lang_temp ( name
varchar(255) NOT NULL, id bigint(20) unsigned NOT NULL, PRIMARY KEY
USING BTREE (id), KEY indice_lang_temp (name)) ENGINE=MyISAM DEFAULT
CHARSET=utf8;" > lang_temp2.log
if [ $? -ne 0 ];
      then
            mysql -h $hst -u $usr -p$pass $1 -e "drop table langlinks;"
            mysql -h $hst -u $usr -p$pass $2 -e "drop table langlinks;"
            mysql -h $hst -u $usr -p$pass $1 -e "drop table lang_temp;"
            echo "error creating table for $s2.lang_temp \n Read
lang_temp2.log"
            exit 1
fi
rm lang_temp2.log
#translating data
echo "translating data $s1.langtemp, it may take a while"
mysql -h $hst -u $usr -p$pass $1 -e "insert into lang_temp (id,name)
SELECT P.id, l.ll_title FROM Page P, langlinks l WHERE
l.ll_from=P.pageId and l.ll_lang = \"$6\"; "> data_temp.log
if [ $? -ne 0 ];
      then
            mysql -h $hst -u $usr -p$pass $1 -e "drop table langlinks;"
            mysql -h $hst -u $usr -p$pass $2 -e "drop table langlinks;"
            mysql -h $hst -u $usr -p$pass $1 -e "drop table lang_temp;"
            mysql -h $hst -u $usr -p$pass $2 -e "drop table lang_temp;"
            echo "error translating data for $s1.langtemp \n Read
data_temp.log"
            exit 1
fi
rm data_temp.log
#
echo "translating data $s1.langtemp, it may take a while"
mysql -h $hst -u $usr -p$pass $2 -e "insert into lang_temp (id,name)
SELECT P.id, l.ll_title FROM Page P, langlinks l WHERE l.ll_from =
P.pageId and l.ll_lang = \"$5\";" > data_temp2.log
if [ $? -ne 0 ];
      then
            mysql -h $hst -u $usr -p$pass $1 -e "drop table langlinks;"
            mysql -h $hst -u $usr -p$pass $2 -e "drop table langlinks;"
            mysql -h $hst -u $usr -p$pass $1 -e "drop table lang_temp;"
            mysql -h $hst -u $usr -p$pass $2 -e "drop table lang_temp;"
            echo "error translating data for $s2.langtemp \n Read
data_temp2.log"
            exit 1
fi
rm data_temp2.log
#
#creating  table translator_$6 for language1
echo "creating  table for $1.translator_$6, please be patient, it may
take a while"
```

```
mysql -h $hst -u $usr -p$pass $1 -e "CREATE TABLE  $1.translator_$6 (
id bigint(20) unsigned default NULL, ll_title varchar(255) default
NULL, id2 bigint(20) unsigned default NULL, PRIMARY KEY  (id,id2),
KEY  pt_id  (id,id2))  ENGINE=MyISAM  DEFAULT  CHARSET=utf8"  >
create_translator.log
if [ $? -ne 0 ];
      then
            mysql -h $hst -u $usr -p$pass $1 -e "drop table langlinks;"
            mysql -h $hst -u $usr -p$pass $2 -e "drop table langlinks;"
            mysql -h $hst -u $usr -p$pass $1 -e "drop table lang_temp;"
            mysql -h $hst -u $usr -p$pass $2 -e "drop table lang_temp;"
            echo "error creating table for $1.translator_$6 \n Read
create_translator.log"
            exit 1
fi
rm create_translator.log
#creating  table translator_$5 for language2
echo "creating  table for $2.translator_$5, please be patient, it may
take a while"
mysql -h $hst -u $usr -p$pass $1 -e "CREATE TABLE  $2.translator_$5 (
id bigint(20) unsigned default NULL, ll_title varchar(255) default
NULL, id2 bigint(20) unsigned default NULL, PRIMARY KEY  (id,id2),
KEY  pt_id  (id,id2))  ENGINE=MyISAM  DEFAULT  CHARSET=utf8"  >
create_translator2.log
if [ $? -ne 0 ];
      then
            mysql -h $hst -u $usr -p$pass $1 -e "drop table langlinks;"
            mysql -h $hst -u $usr -p$pass $2 -e "drop table langlinks;"
            mysql -h $hst -u $usr -p$pass $1 -e "drop table lang_temp;"
            mysql -h $hst -u $usr -p$pass $2 -e "drop table lang_temp;"
            mysql  -h  $hst  -u  $usr  -p$pass  $1  -e  "drop  table
translator_$6;"
            echo  "error  creating  table  for  $1.translator_$5  \n
create_translator2.log"
            exit 1
fi
rm create_translator2.log
#inserting data into translator_$6 for language1
echo "inserting data into $1.translator_$6, please be patient, it may
take a while"
mysql  -h  $hst  -u  $usr  -p$pass  $1  -e  "insert  translator_$6
(id,ll_title,id2) select l.id, l.name, t.id from lang_temp l, $2.Page
t where l.name= t.name; " > insert_translator.log
if [ $? -ne 0 ];
      then
            mysql -h $hst -u $usr -p$pass $1 -e "drop table langlinks;"
            mysql -h $hst -u $usr -p$pass $2 -e "drop table langlinks;"
            mysql -h $hst -u $usr -p$pass $1 -e "drop table lang_temp;"
            mysql -h $hst -u $usr -p$pass $2 -e "drop table lang_temp;"
            mysql  -h  $hst  -u  $usr  -p$pass  $1  -e  "drop  table
translator_$6;"
            mysql  -h  $hst  -u  $usr  -p$pass  $1  -e  "drop  table
translator_$5;"
            echo  "error  inserting  data  for  $1.translator_$6  \n
insert_translator.log"
            exit 1
fi
rm insert_translator.log
#inserting data into translator_$5 for language2
echo "inserting data into $2.translator_$5, please be patient, it may
take a while"
```

```
mysql -h $hst -u $usr -p$pass $2 -e "insert translator_$5
(id,ll_title,id2) select l.id, l.name, t.id from lang_temp l, $1.Page
t where l.name= t.name; " > insert_translator2.log
if [ $? -ne 0 ];
      then
            mysql -h $hst -u $usr -p$pass $1 -e "drop table langlinks;"
            mysql -h $hst -u $usr -p$pass $2 -e "drop table langlinks;"
            mysql -h $hst -u $usr -p$pass $1 -e "drop table lang_temp;"
            mysql -h $hst -u $usr -p$pass $2 -e "drop table lang_temp;"
            mysql -h $hst -u $usr -p$pass $1 -e "drop table
translator_$6;"
            mysql -h $hst -u $usr -p$pass $1 -e "drop table
translator_$5;"
            echo "error inserting data for $2.translator_$5 \n
insert_translator2.log"
            exit 1
fi
rm insert_translator2.log
#mixing data into translator_$6 for language1
echo "mixing data into $1.translator_$6, please be patient, it may
take a while"
mysql -h $hst -u $usr -p$pass $1 -e "insert into translator_$6 (id,
ll_title, id2) select t.id2, p.name, t.id from $2.translator_$5 t,
$2.Page p where p.id=t.id and (t.id, t.id2) not in (select e.id2, e.id
from translator_$6 e);" > mixing_translator.log
if [ $? -ne 0 ];
      then
            mysql -h $hst -u $usr -p$pass $1 -e "drop table langlinks;"
            mysql -h $hst -u $usr -p$pass $2 -e "drop table langlinks;"
            mysql -h $hst -u $usr -p$pass $1 -e "drop table lang_temp;"
            mysql -h $hst -u $usr -p$pass $2 -e "drop table lang_temp;"
            mysql -h $hst -u $usr -p$pass $1 -e "drop table
translator_$6;"
            mysql -h $hst -u $usr -p$pass $1 -e "drop table
translator_$5;"
            echo "error mixing into $1.translator_$6 \n
mixing_translator.log"
            exit 1
fi
rm mixing_translator.log
#mixing data into translator_$5 for language1
echo "mixing data into $2.translator_$5, please be patient, it may
take a while"
mysql -h $hst -u $usr -p$pass $2 -e "insert into translator_$5 (id,
ll_title, id2) select t.id2, p.name, t.id from $1.translator_$6 t,
$1.Page p where p.id=t.id and (t.id, t.id2) not in (select e.id2, e.id
from translator_$5 e);" > mixing_translator2.log
if [ $? -ne 0 ];
      then
            mysql -h $hst -u $usr -p$pass $1 -e "drop table langlinks;"
            mysql -h $hst -u $usr -p$pass $2 -e "drop table langlinks;"
            mysql -h $hst -u $usr -p$pass $1 -e "drop table lang_temp;"
            mysql -h $hst -u $usr -p$pass $2 -e "drop table lang_temp;"
            mysql -h $hst -u $usr -p$pass $1 -e "drop table
translator_$6;"
            mysql -h $hst -u $usr -p$pass $1 -e "drop table
translator_$5;"
            echo "error mixing into $2.translator_$5 \n
mixing_translator2.log"
            exit 1
fi
rm mixing_translator2.log
```

```
#dropping tables
echo "dropping temporal tables"
mysql -h $hst -u $usr -p$pass $1 -e "drop table langlinks;"
mysql -h $hst -u $usr -p$pass $1 -e "drop table lang_temp;"
mysql -h $hst -u $usr -p$pass $2 -e "drop table langlinks;"
mysql -h $hst -u $usr -p$pass $2 -e "drop table lang_temp;"
echo "process completed successfully"
```

12. Bibliografía y Referencias.

- *Documentación Web de instalación de JWPL*. Torsten Zesch , Christof Müller and Iryna Gurevych. Ubiquitous Knowledge Processing Lab, Computer Science Department, Technische Universität Darmstadt. http://www.ukp.tu-darmstadt.de/software/jwpl/documentation/?L=3 .

- *Extracting Lexical Semantic Knowledge from Wikipedia and Wiktionary*. Proceedings of the 6th International Conference on Language Resources and Evaluation, May 2008. Torsten Zesch , Christof Müller and Iryna Gurevych. Ubiquitous Knowledge Processing Lab, Computer Science Department, Technische Universität Darmstadt. Disponible en < http://www.ukp.tu-darmstadt.de/publications/details/?no_cache=1&L=3&pub_id=TUD-CS-2008-4>

- *Sams teach yourself PHP, MySQL and Apache all in one*. Melonie, Julie C. Indianapolis (Indiana) : Sams, 2006. Ed 3. Disponible en web : < http://proquest.safaribooksonline.com/0672328739 >

- *Guía Wiki para Ubuntu*. Disponible en <www.guia-ubuntu.org>

- *Technical Manual for the MediaWiki Software*. Disponible en <http://www.mediawiki.org/wiki/Manual:Contents>

- *Wikipedia Database Download Manual*. Disponible en <http://en.wikipedia.org/wiki/Wikipedia:Database_download>

- *MySQL Avanzado [optimización, copias de seguridad, replicación y equilibrado de carga]* , Zawodny, Jeremy D. Madrid : Anaya Multimedia , 2004. 336 p.

- *La frecuencia léxica y su utilidad en la enseñanza del Español como lengua Extranjera*. Manuel Alvar Ezquerra. Madrid: ASELE. Actas XV, 2004.

- *MySQL 5 : curso profesional de programación de bases de datos.* D'Andrea, Edgar. Barcelona : Inforbook's ; D.L., 2005. 639 p.

- *Apuntes de clase de la asignatura de Ficheros y Bases de Datos*. Profesor David Rodríguez Alvarez. LABDA. Universidad Carlos III de Madrid. Disponible en <http://basesdatos.uc3m.es>

- *WWW::Wikipedia - Automated interface to the Wikipedia*. Brian Cassidy, Ed Summers. 2006. Disponible en < http://search.cpan.org/~bricas/WWW-Wikipedia-1.95>

- *Parse::MediaWikiDump - Tools to process MediaWiki dump files.* Tyler Riddle 2006. Disponible en http://search.cpan.org/~triddle/Parse-MediaWikiDump/

- *Wikipedia Preprocessor (WikiPrep).* Evgeniy Gabrilovich. Disponible en < http://www.cs.technion.ac.il/~gabr/resources/code/wikiprep/>